VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MATEMATIKY
INSTITUTE OF MATHEMATICS

DETEKCE A VIZUALIZACE SPECIFICKÝCH RYSŮ V MRAČNU BODŮ
DETECTION AND VIZUALIZATION OF FEATURES IN A POINT CLOUD

DIZERTAČNÍ PRÁCE
DOCTORAL THESIS

AUTOR PRÁCE — **Mgr. Jiří Kratochvíl**
AUTHOR

ŠKOLITEL — **Mgr. Jana Procházková, Ph.D.**
SUPERVISOR

BRNO 2018

Abstrakt
Mračno bodů je neuspořádaná množina bodů popsaných souřadnicemi (x, y, z), která reprezentuje reálný objekt. Tyto body jsou získány prostřednictvím 3D skenovacích technologií, například LIDAR (Light Detection And Ranging) nebo pomocí současných 3D skenerů. Získaná mračna bodů jsou pak využívána v široké škále odvětví lidské činnosti, jako například strojní či reverzní inženýrství, rapid prototyping, biologie, nukleární fyzika nebo virtuální realita.

Tato disertační práce přispívá k vývoji metod pro detekci bodů na specifických rysech v mračnu bodů, což se v anglické literatuře označuje pojmem feature detection. Dále k vývoji metod jejich vizualizací prostřednictvím prokladu křivek, také známé pod pojmeme curve fitting. Feature, či specifický rys, je významná část objektu, kterou se snažíme popsat matematickým modelem (např. rovinou, přímkou či křivkou). Obzvláště body na ostrých hranách jsou pro současné metody problematické, a proto se věnujeme jejich detekci. V disertační práci je popsán nový algoritmus, který automaticky a s velkou přesností určuje tyto body. Jejich vizualizace je potom zajištěna proložením křivkou, kde byla doplněna nová metoda váhování pro přesnější výsledky. Všechny navržené postupy byly otestovány na reálných datech a srovnány s dosavadními publikovanými metodami.

Summary
The point cloud is an unorganized set of points with 3D coordinates (x, y, z) which represents a real object. These point clouds are acquired by the technology called 3D scanning. This scanning technique can be done by various methods, such as LIDAR (Light Detection And Ranging) or by utilizing recently developed 3D scanners. Point clouds can be therefore used in various applications, such as mechanical or reverse engineering, rapid prototyping, biology, nuclear physics or virtual reality.

Therefore in this doctoral Ph.D. thesis, I focus on feature detection and visualization in a point cloud. These features represent parts of the object that can be described by the well–known mathematical model (lines, planes, helices etc.). The points on the sharp edges are especialy problematic for commonly used methods. Therefore, I focus on detection of these problematic points. This doctoral Ph.D. thesis presents a new algorithm for precise detection of these problematic points. Visualization of these points is done by a modified curve fitting algoritm with a new weight function that leads to better results. Each of the proposed methods were tested on real data sets and compared with contemporary published methods.

Klíčová slova
počítačová grafika, mračno bodů, PCL, LIDAR, ostré hrany, feature, detekce, vizualizace, region growing, segmentace, triangulace, filtrování, curve fitting, prokládání křivek

Keywords
computer graphics, point cloud, PCL, LIDAR, sharp feature, feature, detection, visualization, region growing, segmentation, triangulation, filtration, curve fitting

KRATOCHVÍL, J.*Detekce a vizualizace specifických rysů v mračnu bodů.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2018. 100 s. Vedoucí diplomové práce Mgr. Jana Procházková, Ph.D.

Prohlašuji, že jsem disertační práci na téma „Detekce a vizualizace specifických rysů v mračnu bodů“ vypracoval samostatně a s použitím uvedené literatury a pramenů.

Mgr. Jiří Kratochvíl

Tímto bych chtěl poděkovat paní Mgr. Janě Procházkové, Ph.D. za užitečné rady, výpomoc a ochotu během zpracovávání mé disertační práce a během celé doby mého doktorského studia na Ústavu matematiky FSI VUT v Brně. Dále bych zde chtěl poděkovat panu prof. RNDr. Miroslavu Druckmüllerovi, CSc. a prof. RNDr. Josefu Šlapalovi, CSc. za možnost zpracování tohoto tématu, které mi otevřelo nové obzory v oblasti počítačové grafiky. V neposlední řadě chci poděkovat panu Ing. Davidu Procházkovi, Ph.D. za možnost účasti na přednáškách věnovaných objektovém programování v jazyce C++ na Mendlově Univerzitě v Brně. Velký dík patří také mé rodině za morální podporu.

Mgr. Jiří Kratochvíl

Obsah

1. Úvod

Od 80. let 20. století dochází k vývoji zobrazovacích metod počítačové grafiky, které využívají jako elementární prvky body. Způsobům zobrazování 3D objektů pomocí bodů se věnují například Levoy a Whitted [37]. Pro tuto problematiku byla v roce 2001 vytvořena softwarová knihovna *PCL* (*Point Cloud Library*) (Rusu a Cousins [53]). PCL je ohromný *open-source, cross-platform* projekt skýtající nepřeberné množství optimalizovaných algoritmů a funkcí pro zpracování a vizualizaci mraků bodů. Na jeho vývoji se mimo jiné podílí i známé firmy, jako například Google, Toyota, Leica či nVidia. V současné době je knihovna PCL ve verzi 1.8.1 a dokumentace je dostupná v [17].

Mračno bodů, jakožto prostředek k vytvoření digitálního 3D modelu reálného předmětu, je často využíváno ve strojírenských odvětvích, například v oblastech reverzního inženýrství nebo rapid prototyping. Další uplatnění nalézají mračna bodů v různých oblastech přírodních věd jako: geografie a kartografie, dále také lékařství, bezpečnostní systémy či virtuální realita.

Disertační práce vznikla na podněty Ústavu konstruování Fakulty strojního inženýrství VUT v Brně, kdy při rekonstrukci mračen bodů nasnímaných předmětů dochází při použití standardních funkcí dostupných v příslušném softwaru k chybám. Chyby se vyskytují především v místech ostrých hran a rohů, která se v angličtině označují jako *sharp feature*. V těchto místech se modely musí ručně opravovat, což je časově dosti náročné a snižuje se tím efektivita práce. Vznikla zde potřeba vytvoření algoritmu, který tyto body automaticky vyhledá.

Feature (do češtiny lze přeložit jako *specifický rys*) je významná část objektu, kterou se snažíme popsat matematickým modelem, například rovinou, přímkou či křivkou. Detekování rovinných bodů není v dnešní době problém, avšak body v oblasti ostrých hran či rohů, způsobují při použití současných standardních metod potíže. Tato práce se věnuje detekci a vizualizaci bodů na těchto problematických místech a tím přispívá k vývoji metod v této oblasti. Navržená metoda je zcela unikátní a svým novým způsobem detekce poskytuje srovnatelné výsledky jako současné publikované metody.

Text disertační práce je rozdělen do dvou hlavních celků: teoretické základy a výsledky. V první části jsou uvedeny informace o mračnu bodů – jeho získávání, zpracování a ukládání, dále je zde popsána knihovna PCL a také nejčastěji používané křivky včetně metod jejich prokladu (*curve fitting*) pro vizualizaci hledaných bodů. Ve druhé části je popsána nová metoda detekce bodů na specifických rysech v mračnu bodů, upravený algoritmus prokladu b-spline křivkou a také srovnání výsledků navržených metod se současnými dostupnými algoritmy.

Disertační práce je napsána v sázecím systému LaTeX ilustrační obrázky vytvořeny v programu Adobe Illustrator a Photoshop CC6, pomocné 3D modely jsou vytvořeny v programu Blender, grafy vytvořeny pomocí Microsoft Excel a MATLAB 2017b. Navržené algoritmy jsou naprogramovány v jazyce C++ v prostředí Microsoft Visual Studio Community 2015 (v. 14) za podpory PCL verze 1.8.1.

2. Cíle práce

Trojrozměrné skenování a získaná mračna bodů jsou významnými prostředky ve strojním inženýrství, speciálně v oblasti reverzního inženýrství. Známé metody a algoritmy jsou schopné z mračna bodů vytvořit virtuální 3D model. V místech ostrých hran nebo dalších specifických rysech skenovaného objektu vznikají nepřesnosti, které jsou způsobeny principem těchto metod. Cílem této práce je přispět k vývoji algoritmů a metod pro detekci bodů na těchto specifických rysech, a tím zaručit lepší a přesnější reprezentaci daného objektu. Ve své práci se zaměřuji na detekci a vizualizaci bodů na tzv. *sharp feature* (do češtiny lze přeložit jako *specifický rys, vlastnost*). Jedná se o část objektu, která vykazuje specifické vlastnosti vzhledem k celkovému objektu, jmenovitě ostrá hrana či roh, a kterou lze popsat matematickým modelem. Cíle disertační práce jsou následující.

1. **Studium problematiky mračen bodů.**
   Zpracování přehledu základních metod a postupů pro pořízení, zpracování a vizualizaci mračna bodů.

2. **Vytvoření nové metody pro vyhledávání bodů na specifických rysech v mračnu bodů a její algoritmizace.**
   Navržení a otestování algoritmu pro detekci bodů na specifických rysech využitím dostupných a modifikovaných metod knihovny PCL. Těmito specifickými rysy uvažujeme především ostré hrany těles, které můžeme prokládat prostorovými křivkami (přímkami, šroubovicemi a dalšími).

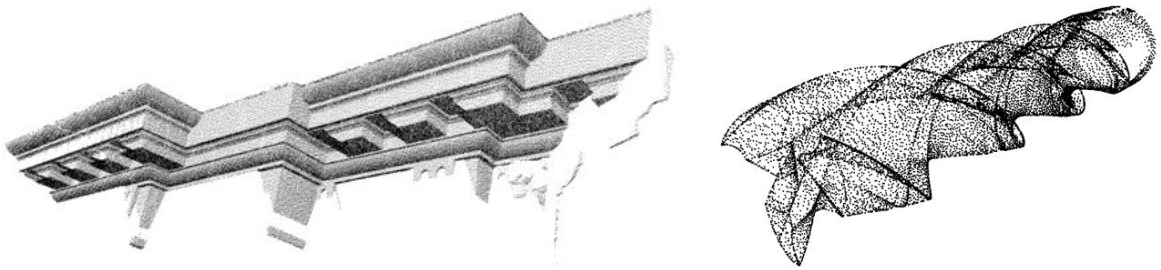
3. **Vizualizace bodů na specifických rysech v mračnu bodů.**
   Grafické znázornění nalezených bodů pomocí jejich matematických modelů prostřednictvím knihoven PCL a VTK. Jedná se o metodu prokladu křivek – tzv. *curve fitting* s následnou vizualizací v programu PCLVisualizer.

3. Teoretické základy

Následující kapitola shrnuje fundametální pojmy, metody a principy pro práci s mračny bodů a je rozdělena do čtyř hlavních částí: (i) mračno bodů, (ii) point cloud library a vizualizační metody, (iii) křivky používané pro vizualizaci, (iv) prokládání dat vhodnými křivkami.

3.1. Mračno bodů

Mračno bodů (v angličtině *point cloud*), je neuspořádaná množina bodů ve dvou nebo třídimenzionálním prostoru, přičemž každý z nich je dán svými souřadnicemi v daném systému souřadnic. Poloha každého bodu P v prostoru je jednoznačně určena souřadnicemi (x, y), případně (x, y, z). Navíc musí být kvantifikovatelná (je definována jednotka systému) a musí být definována metrika [28], pomocí níž lze měřit vzdálenost mezi body.



(a) (b)

Obrázek 3.1: Mračno bodů (a) část místnosti s 108 104 body, (b) fréza s 15 360 body.

Mračno bodů představuje digitální podobu – vizualizaci reálného předmětu (od malých strojních součástí až po celá města či velké části zemského terénu, viz obr. 3.1), která vzniká za pomoci nepřeberného množství zařízení a skenovacích systémů. Mezi nejčastěji používané patří laserové snímání (viz kap. 3.1.1 str. 6), jehož principu využívá letecké laserové snímání, pozemní skenování, nebo například ruční skenery či zařízení Microsoft Kinect. Získaný sken v podobě mračna bodů je uložen v příslušném datovém formátu (viz kap. 3.1.2) a je zpracováván velkou škálou metod a algoritmů (viz kap. 3.1.3 str. 16).

Digitální podoba objektu umožňuje jeho následné využití v širokém spektru lidské činnosti prostřednictvím výpočetní techniky, která je v současnosti neodmyslitelnou součástí každodenního života. S mračny bodů se můžeme setkat například v oblasti reverzního inženýrství, kde prostorový model umožňuje úpravu a následnou výrobu předmětu, u kterého není k dispozici technická dokumentace. Dále například v kartografii a geografii, kde letecky nasnímané oblasti zemského povrchu slouží k vytvoření digitálních map dané lokace, které pak využívá globální poziční systém GPS či například služba Google Mapy [43]. Další užitečné využití mračna bodů je bezpochyby v lékařství, kde je například možné vytvoření protetické pomůcky dle potřeb individuálních pacientů. V neposlední řadě se s mračny bodů můžeme setkat v odvětví virtuální reality a herního či filmového průmyslu, kde nasnímané objekty představují co nejpřesnější reprezentaci reality.

3.1.1. Metody získávání mračna bodů

Velká většina zařízení k vytvoření mračna bodů daného předmětu využívá laserového snímání, nebo-li *LIDAR* (*Light Detection And Ranging*) [55]. Snímací zařízení zpravidla obsahuje vysílač a snímač. Vysílač vytváří laserový paprsek, který dopadá na předmět zájmu. Paprsek se od objektu odrazí a je následně zachycen ve snímači, který každému paprsku přiřadí koordináty v daném souřadném systému. Jestliže je ve snímacím zařízení zabudovaná i RGB kamera, přiřadí se konkrétnímu bodu i barva ve viditelném nebo infračerveném spektru. Obecně lze LIDAR rozdělit na letecký a pozemní.

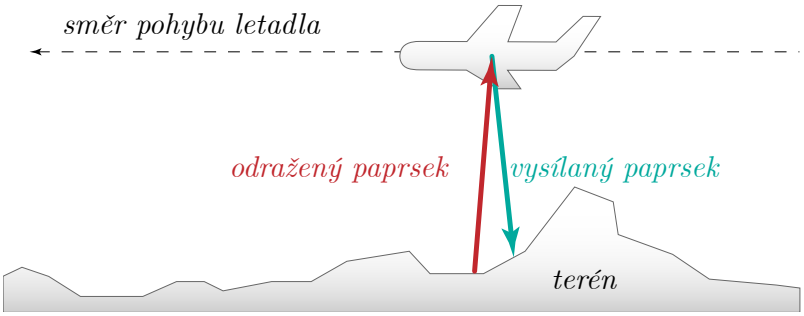
Skenovací zařízení lze rozdělit do dvou skupin podle způsobu snímání, a to *triangulační* či *time-of-travel*. V prvním případě se na objekt promítne mřížka tvořená rovnoměrně rozmístěnými světelnými body. Ta je následně snímána kamerou a podle její deformace vzhledem k referenční rovině se vytvoří model. V druhém případě se měří čas, za který světelný paprsek urazí cestu (již zmíněný time-of-travel) mezi vysílačem, objektem a snímačem. Následně se tato doba s ohledem na směr paprsku přepočte na souřadnice daného bodu. Zařízení tohoto typu vždy vysílá několik paprsků současně, obvykle v podobě horizontální či vertikální přímky, které se po předmětu pohybují v určeném směru danou rychlostí. Snímač pak v nastavených intervalech zachytává světelné body těchto přímek a měří jejich time-of-travel. Získaný model – obraz se v obou případech označuje jako tzv. *range image*.

Jak je zřejmé, snímání se provádí prostřednictvím světelných – laserových paprsků, které mohou být ovlivněny řadou faktorů a měření může být zkreslené. Jde například o zastínění skenovaného objektu, kdy se mezi předmětem a skenerem nachází překážka, nebo je část objektu zastíněna sama sebou. S touto situací se můžeme setkat například při leteckém snímkování měst, kdy paprsek nepronikne hustou zelení či mosty. V případě, že je objekt zastíněn sám sebou, je nutné vytvořit sérií skenů z různých pohledů, které se posléze spojí v celkový model. Měření může být mírně zkreslené také díky změnám v atmosféře (teplota, vlhkost, znečištění), avšak tento faktor lze omezit vhodným naplánováním snímacího procesu. V případě zařízení typu time-of-travel se může vyskytnou odchylka při práci s předměty, jejichž povrch má relativně velkou světelnou odrazivost či absorpci. V praxi je proto nutná úprava skenovaného objektu, která se většinou provádí antireflexním nástřikem. V neposlední řadě může nastat zkreslení či úplné znehodnocení měření v případě vibrací, které způsobují změnu ve stacionární poloze skenovacího zařízení či objektu.

V následující části jsou popsány metody leteckého či pozemního snímání, dále ruční skenery a také zařízení Microsoft Kinect.

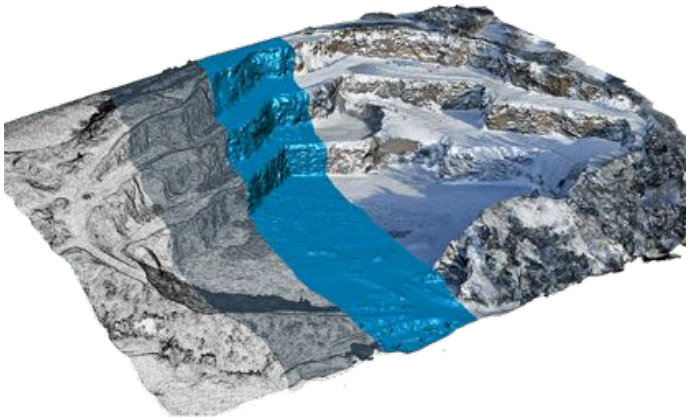
Letecké skenování ALS

Airborne Laser Scanning (*ALS*) [66], či letecké laserové skenování, je snímací metoda pro mapování zemského povrchu. Měřící systém je instalován na palubě letadla a snímá terén pod ním v určeném čase v průběhu letu. ALS systém je vybaven třemi hlavními komponentami: interní měřící jednotka (Internal Measurement Unit, IMU); přijímač GPS (Global Positioning System) signálu; a jednotku laserového skeneru. Měřící systém zaznamenává čas a energii odraženého laserového paprsku, který skener vysílá (viz obr. 3.2). Tyto hodnoty společně s parametry letu jsou použity pro vytvoření digitálního 3D modelu daného terénu, který je umístěn v georeferenčním souřadném systému.



Obrázek 3.2: Princip skenování ALS.

Z naměřených dat může být posléze vytvořen *Digital Elevation Model* (*digitální výškový model, DEM*) [39], který může být reprezentován jako část *Digital Terrain Model* (*digitální model terénu, DTM*) nebo *Digital Surface Model* (*digitální model povrchu, DSM*). DTM obsahuje informace o terénu (kopce, údolí, propasti aj.) a DSM přidává data o objektech na povrchu (automobily, domy, vegetace apod.) (viz obr. 3.3). Modely měst lze například využít k detekci střech vhodných pro instalaci fotovoltaických panelů [2].



Obrázek 3.3: DSM, upraveno, G4D [online]. [cit. 4.3.2017].
Dostupný na http://www.g4d.cz/en/digital-3d-models/digital-terrain-surface-models.

Pozemní skenování

Pozemní či mobilní skenovací systémy jsou určeny pro vytvoření virtuálního modelu menší oblasti, než je v případě ALS. Může se například jednat o detailnější model městských budov, dopravní infrastruktury (křižovatky, tunely, parky aj.) nebo také historických památihodností či interiérů [5].

Pozemní snímkování lze podle funkce rozdělit na *stacionární* a *mobilní*. V prvním případě je skenovací systém umístěn na pevném – stacionárním místě a jeho rotací kolem vlastní osy se postupně vytváří výsledný sken okolního prostředí. Tato varianta je vhodná pro snímání interiérů, menších částí otevřeného prostoru (náměstí, amfiteátry, nádvoří apod.), jednotlivých předmětů (strojní součásti, spotřební zboží, rostliny aj.) či osob atd. Pro každou oblast využití existuje řada vhodných skenerů, jejichž měřící vlastnosti (rozměry, hmotnost, rozlišení, měřící objemy aj.) jsou vyhovující. Systém je podobně jako v případě ALS vybaven interní měřící jednotkou (IMU), globálním pozičním systémem

(GPS) a laserovým skenerem. Ve speciálních případech je součástí i digitální kamera pro pořízení obrazového záznamu.

Druhá varianta – mobilní skenování je pohyblivou verzí stacionárních skenovacích systémů. Snímací jednotka je obvykle umístěna na střeše automobilu a v průběhu jízdy se vytváří virtuální model okolní oblasti. Oproti stacionárním typům se zde kombinuje několik laserových skenerů, které jsou orientovány do více stran (nahoru, dolů a do stran). S výhodou se této možnosti využívá pro skenování dopravní infrastruktury včetně okolního prostředí. Průkopníkem mobilního skenování je bezpochyby společnost Google, která svá data implementovala do vlastních Google Maps [43].

Mezi často používané pozemní stacionární skenery patří například systémy firmy Leica, ATOS, FARO (viz obr. 3.4).

(a) (b) (c)

Obrázek 3.4: (a) Leica ScanStation P40, (b) ATOS triple scan, (c) FARO focus 3D.

Ruční skenery

V oblasti 3D skenerů nastal v posledních letech velký vývoj. Kromě průmyslových skenovacích systémů se do popředí dostávají i kompaktnější a cenově dostupnější varianty. Mezi cenově nejpříznivější ruční skenery patří například: Sense Cubify, iSense Cubify, Occipital Structure Sensor nebo XYZprinting 3D scanner (obr. 3.5). iSense Cubify a Occipital Structure Sensor jsou vytvořeny speciálně pro tablety iPad a jedná se v podstatě o stejné produkty.

(a) (b) (c) (d)

Obrázek 3.5: (a) Sense Cubify, (b) XYZprinting 3D scanner, (c) iSense Cubify, (d) Occipital Structure Sensor 3D.

Microsoft Kinect

Zařízení Kinect společnosti Microsoft je pohybový senzor určený pro herní konzole Xbox. První verze s označením Kinect V1 (obr. 3.6(a)) byla vydána v roce 2010 při představení nové herní konzole Xbox 360. O tři roky později, v roce 2013, Microsoft uvedl na trh vylepšenou konzoli Xbox One společně se zdokonaleným senzorem Kinect V2 (obr. 3.6(b)).

(a) (b)

Obrázek 3.6: (a) Kinect V1, (b) Kinect V2.

Kinect je vybaven RGB kamerou, hloubkovým senzorem a mikrofonem. Původní verze V1 pracovala na principu technologie PrimeSense [30], [33], která využívá principu triangulace (viz kap. 3.1.1 str. 6). Současná vylepšená varianta V2 pracuje na principu time-of-travel a tím dokáže přesněji snímat tvar a pohyb objektů. Nashromážděná data v podobě mračna bodů se zpracovávají pro širokou škálu využití s herní konzolí.

Díky cenové dostupnosti a přesnosti se Kinect dostal i do povědomí vědecké veřejnosti, která jej může využít pro vlastní výzkumy. Původně produkt určený pro komerční využití v herním průmyslu se tedy v současnosti stává výzkumnou pomůckou a také levnější variantou industriálních 3D skenerů. Na podzim roku 2017 byla výroba senzoru Kinect firmou Microsoft zastavena. Senzor totiž ztrácel popularitu a dnešní konzole s ním již nepočítají.

ATOS Compact Scan

FSI VUT v Brně má od roku 2013 k dispozici 3D skener ATOS Compact Scan 2M od německého výrobce GOM mbH, Braunschweig (obr. 3.7). Skener byl pořízen v rámci projektu **Prostorový skener pro výuku reverzního inženýrství, FRVŠ č. 1393/2013** na Ústav automobilního a dopravního inženýrství, a v současnosti je umístěn na 3D pracovišti FSI VUT v Brně. Hlavním řešitelem projektu byl prof. Ing. Václav Píštěk, DrSc. Na tomto zařízení bylo nasnímáno několik modelů, se kterými pracuji ve své disertační práci.

Obrázek 3.7: ATOS Compact Scan 2M.

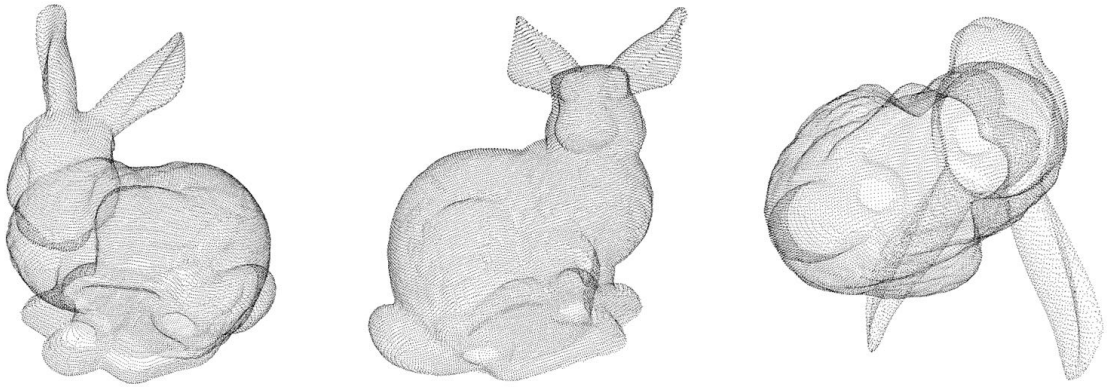
Tento 3D skener používá tři různé měřicí objemy (125x90x90mm, 250x190x190mm, 1000x750x750mm) a pracuje na ATOS projekční technologii modrého světla se dvěma

digitálními kamerami. Světelný zdroj promítá modrým světlem mřížku na snímaný objekt a obě kamery zaznamenávají její změnu vzhledem k referenční podobě. Jde tedy o triangulační skenovací princip. Na skenovaném předmětu jsou navíc nalepeny referenční body a tím je zaručeno správné propojení skenů z jednotlivých pohledů do výsledného modelu. Skener je vybaven ramenem s nastavitelnou výškou a také umožňuje rotaci skenovací hlavy pro zaměření sledovaného objektu. Snímaný objekt je možné uložit na otáčivý stůl, který usnadňuje skenovací proces. Skener je vybaven ATOS Professional scanning software verze V.7.5 SR2, ve kterém se skenovaná data mohou následně zpracovat a dále ukládat či exportovat do standardních formátů pro mračna bodů.

3.1.2. Datové formáty pro ukládání mračen bodů

V této kapitole jsou v jednoduchosti popsány nejpoužívanější formáty souborů pro ukládání dat mračen bodů, jmenovitě *OBJ* [46], [63]; *PCD* [16]; *PLY* [50]; *STL* [56] a *X3D* [68]. U každého typu je uvedena krátká ukázka zdrojového souboru.

Některé výzkumné instituce, univerzity či softwarové firmy poskytují svá 3D data ve formě mračen bodů v internetových depozitářích. Jde například o skeny laboratoří Autodesk či Leica Geosystems a v současnosti nejpoužívanější modely pro výzkumné potřeby jsou 3D skeny Stanfordovy univerzity v Kalifornii, USA. Mezi tyto skeny patří například tzv. *Stanford bunny* (obr. 3.8).



(a) (b) (c)

Obrázek 3.8: Stanford bunny – model zajíce s 35 947 body.

Přístup k těmto modelům umožňuje vývoj nových metod i v místech, kde nejsou k dispozici 3D skenery pro jejich vytvoření. Navíc použití stejných modelů také zajišťuje snadnější porovnání experimentálních metod s metodami ostatních autorů z celého světa.

OBJ

- *OBJect files*
- vytvořen pro popis 3D objektů v Advanced Visualizeru (80. léta 20. století) firmy Wavefront Technologies
- popisuje vrcholy včetně normál ploch v těchto bodech, stěny včetně normál, křivky a také například barvu či texturu povrchu

- struktura souboru je poměrně komplexní
- zjednodušená forma:
  - hlavička
  - seznam bodů
  - seznam souřadnic textur
  - seznam normál ploch v bodech
  - seznam stěn, křivek, ploch a další
- ukázka souboru:

```
# Vertices: 7
# Points: 0
# Lines: 0
# Faces: 5
# Materials: 1

o 1

# Vertex list

v -0.5 -0.5 0.5
v -0.5 -0.5 -0.5
v -0.5 0.5 -0.5
v -0.5 0.5 0.5
v 0.5 -0.5 0.5
v 0.5 -0.5 -0.5
v 0.5 0.5 -0.5

# Point/Line/Face list

usemtl Default
f 4 3 2 1
f 2 6 5 1
f 3 7 6 2
f 8 7 3 4
f 5 8 4 1

# End of file
```

PCD

- *Point Cloud Data*
- vytvořený v rámci knihovny PCL [53]
- popisuje pouze souřadnice bodů v mračnu bodů
- struktura:
  - hlavička
  - souřadnice bodů
- ukázka souboru:

```
# .PCD v0.7 - Point Cloud Data file format
VERSION .7
FIELDS x y z
SIZE 1 1 1
TYPE F F F
COUNT 1 1 1
WIDTH 16
HEIGHT 1
POINTS 16
DATA ascii
-1.0 -1.0 -1.0
-1.0 -1.0 1.0
-1.0 1.0 -1.0
-1.0 1.0 1.0
1.0 -1.0 -1.0
1.0 -1.0 1.0
1.0 1.0 -1.0
1.0 1.0 1.0
-1.0 0.8181818127632141 -1.0
-1.0 0.6363636255264282 -1.0
-1.0 0.45454543828964233 -1.0
-1.0 0.27272725105285645 -1.0
-1.0 0.09090906381607056 -1.0
-1.0 -0.09090910851955414 -1.0
-1.0 -0.27272728085517883 -1.0
-1.0 -0.4545454680919647 -1.0
```

PLY

- *Polygon File Format*, případně *Stanford Triangle Format*
- 1994, Greg Turk, Standford University
- vytvořen za účelem ukládání dat z 3D skenerů
- popisuje vrcholy, stěny a také barvu či normálové vektory
- struktura:
  - hlavička
  - seznam bodů
  - seznam stěn
  - seznam dalších prvků objektu
- ukázka souboru:

```
ply format ascii 1.0
comment this file is a cube
element vertex 8
property float x
property float y
property float z
element face 6
property list uchar int vertex_index
end_header
0 0 0
0 0 1
0 1 1
0 1 0
1 0 0
1 0 1
1 1 1
1 1 0
4 0 1 2 3
4 7 6 5 4
4 0 4 5 1
4 1 5 6 2
4 2 6 7 3
4 3 7 4 0
```

STL

- *STereo Lithography*
- 1987, Albert Consulting Group
- vytvořen pro systémy rapid prototyping – 3D tiskárny
- popisuje objekt pomocí trojúhelníkové sítě s příslušnými body a normálovými vektory
- struktura:
  - začíná slovem „solid“ zpravidla následovaným názvem souboru, autorem apod.
  - normála trojúhelníku
  - tři body určující tento trojúhelník
  - další trojúhelníky určené třemi body a normálou
  - končí slovem „endsolid“
- ukázka souboru:

```
solid
⋮
facet normal 0.0 0.0 1.0
outer loop
vertex 1.0 1.0 0.0
vertex -1.0 1.0 0.0
vertex 0.0 -1.0 0.0
endloop
endfacet
⋮
endsolid
```

X3D

- *eXtensible 3D*
- bezplatný ISO standard pro ukládání 3D scén
- využívá *XML* (*eXtensible Markup Language*) zápis
- popisuje celkovou scénu včetně objektů pro následnou vizualizaci
- zjednodušená struktura:
  - hlavička
  - popis scény
- ukázka souboru:

```
<?xml version="1.0"encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.2//EN"
"http://www.web3d.org/specifications/x3d-3.2.dtd»

<X3D profile="Interchange"version="3.2"
xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
xsd:noNamespaceSchemaLocation="http://www.web3d.org/
specifications/x3d-3.2.xsd»
<Scene>
<Background DEF="WO_World"
groundColor="0.051 0.051 0.051"
skyColor="0.051 0.051 0.051"
/>
<Shape>
<Appearance>
</Appearance>
<IndexedFaceSet coordIndex="0 1 2»
<Coordinate point="0 0 0 1 0 0 0.5 1 0"/>
</IndexedFaceSet>
</Shape>
<Viewpoint DEF="CA_Camera"
centerOfRotation="0 0 0"
position="-0.00 -0.00 0.00"
orientation="-0.00 -0.47 -0.88 0.00"
fieldOfView="0.858"
/>
</Scene>
</X3D>
```

3.1.3. Metody zpracovávání mračen bodů

Mračno bodů, získané pomocí skenovacích metod (kap. 3.1.1 str. 6) a uložené v příslušném formátu (kap. 3.1.2 str. 10), se následně zpracovává různými metodami. Lze je obecně rozdělit do dvou kategorií: *preprocesové* a *postprocesové*. V této kapitole jsou popsány některé často používané algoritmy těchto dvou kategorií. Preprocesové algoritmy slouží k úpravě „surových“ (původních) dat, aby se s nimi mohlo lépe pracovat. Patří sem například: kd–tree, registrace či filtrace. Postprocesové metody slouží k dalšímu zpracovávání dat. Jde například o segmentaci, triangulaci, výpočet normálových vektorů či křivosti. Většina z uvedených metod využívá statistické výpočty jako například modus, medián či kovarianční matici [60].

kd–tree

Kd-tree (*k-dimensional tree*) [9], nebo-li *k-dimenzionální strom*, je preprocesový algoritmus, který slouží k vytvoření organizované struktury daného mračna. Tato struktura se označuje slovem *tree* čili *strom* vzhledem k obdobné podobě.

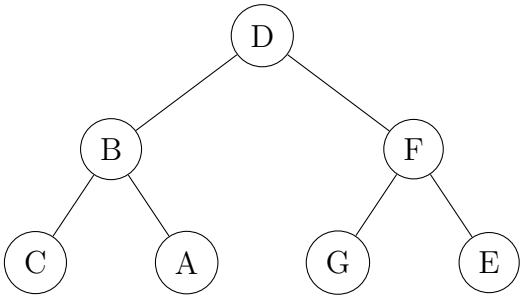
Algoritmus funguje ve dvou základních krocích. V prvním vybere libovolnou osu ze souřadného systému. V druhém kroku spočte medián hodnot v této ose, a v tomto bodě data rozdělí na dvě části. V dílčích částech se opět zvolí souřadná osa (odlišná od předchozí volby), spočte se medián a data se znovu rozdělí. Tak se postupuje, dokud se nezpracují všechny body.

Při výpočtu mediánu mohou nastat dvě situace: (i) medián leží přesně v hodnotě souřadnic některého bodu; (ii) medián leží mezi hodnotami souřadnic dvou bodů. V první situaci se vybere přímo bod s touto souřadnicí. V druhém případě se určí, zda se budou vybírat vyšší či nižší hodnoty, a tak se bude postupovat i v dalších výskytech tohoto jevu. V této situaci mohou vznikat tzv. *unbalanced trees* neboli *nevyvážené stromy*, kdy jsou ve stejné úrovni v dílčích větvích různé počty bodů. Existuje však skupina algoritmů, které dokáží stromy opět vyvážit [34].

Prvotní dělící bod představuje *kmen* pomyslného stromu, další body rozdělení *větve* a všechny ostatní body, ve kterých se již nedělí jsou *listy*, viz následující příklad.

Příklad 3.1. Mějme 7 bodů v rovině se souřadnicemi: A[1,7], B[2,6], C[3,5], D[4,4], E[5,3], F[6,2], G[7,1]. Rovina má dimenzi 2, tedy $k = 2$ a souřadné osy jsou x, y. Vytvoříme tedy 2d-strom zadaných bodů. Zvolme dělicí osu x, ve které je medián hodnot roven 4. Tedy bod D je první dělicí bod. Vlevo budou body A, B, C a vpravo E, F, G. Nyní se provede rozdělení v levé a pravé části podle druhé osy y. V levé polovině (A, B, C) je medián y-ových hodnot roven 5, a tedy bod B bude bod rozdělení. Bod C s menší souřadnicí y zakreslíme vlevo a bod A vpravo. V polovině napravo od bodu D je medián y-ových hodnot roven 2, tudíž rozdělíme v bodě F. Bod G vykreslíme vlevo a bod E vpravo. Prošli jsme všechny zadané body a strom je tedy hotov (viz obr. 3.9).

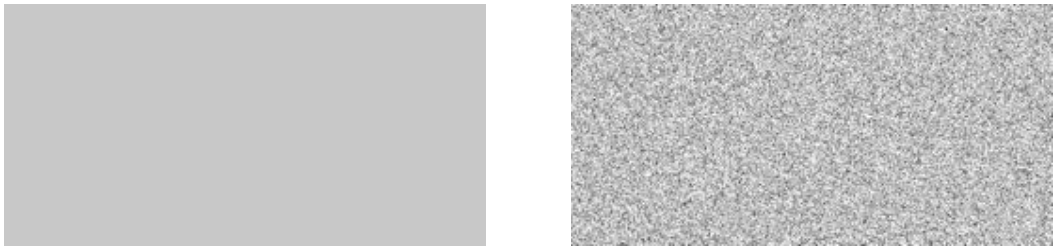
S tímto algoritmem, jako preprocesem pracují téměř všechny uvedené metody. Avšak nejčastější implementace je pro zjištění k-nejbližších sousedních bodů zvoleného bodu.



Obrázek 3.9: 2d-strom zadaných bodů.

Filtrace

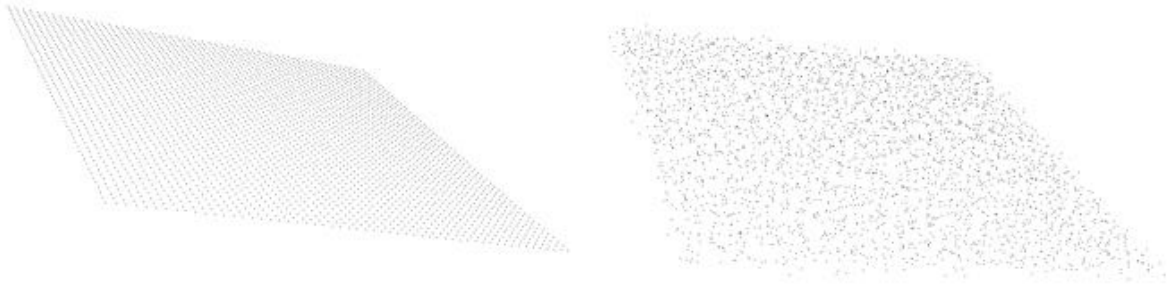
Mračna bodů nasnímaných předmětů zpravidla obsahují nepřesné, či chybné body, které jsou způsobeny různými faktory během skenovacího procesu. Tyto nepřesnosti se označují jako *šum*. Šum lze částečně eliminovat kalibrací skeneru, či snímáním za kontrolovaných podmínek. Šumu však nikdy zcela nepředejdeme. Pro jeho ilustraci použijeme podobnou situaci s oblasti fotografie. Zde se projevuje, jako pixely s výrazně odlišnou barvou či intenzitou než jejich okolí. Typickým případem jsou jednobarevné plochy (viz obr. 3.10).



(a) (b)

Obrázek 3.10: (a) obraz bez šumu, (b) obraz se šumem.

V mračnu bodů se šum projevuje jako body s odlišnou polohou, než je poloha většiny okolních bodů (obr. 3.11). Tyto vadné body mohou způsobit zkreslení výsledků při dalším zpracovávání dat, a proto je nutné jejich počet snížit. Problém šumu a dalších nežádoucích aspektů řeší tzv. *filtrace*. Filtrační metody procházejí vstupní data a v závislosti na určených parametrech body buď ponechají, nebo je odstraní. Filtrace může být víceprůchodová, kdy se data v každém dalším kroku prochází s přísnějšími kritérii, dokud není dosaženo požadovaného výsledku. Filtrované mračno bodů pak slouží jako vstup pro další metody. Přehled současných metod filtrace mračen bodů je možné nalézt v [44].



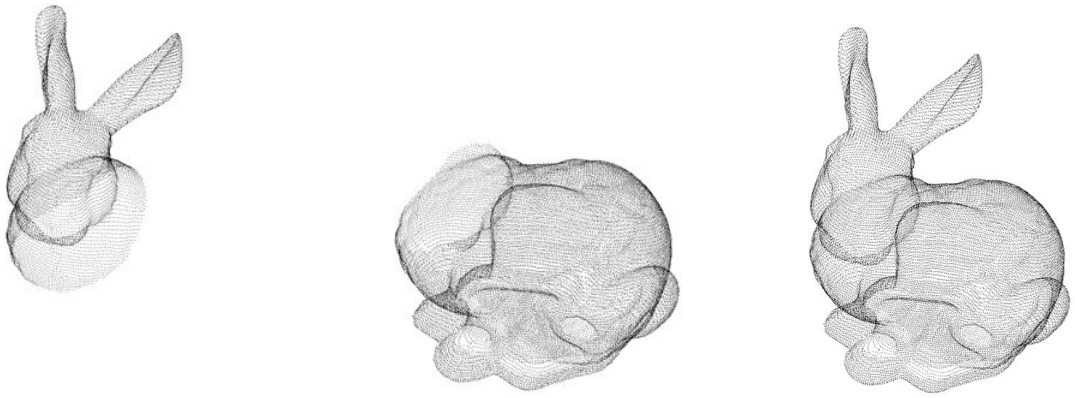
(a) (b)

Obrázek 3.11: Mračno bodů roviny (a) bez šumu, (b) 6% šum.

Registrace

Jednou z problematik při 3D skenování složitějších či rozměrově větších objektů je potřeba vytvoření několika dílčích snímků z různých pohledů. Zpravidla se jedná o dvě a více mračen bodů, které se následně sloučí v jeden celek pomocí metody zvané *registrace* (obr. 3.12).

Registrační algoritmy nejprve vyhledávají odpovídající si body ve vstupních mračnech a posléze nalézají vhodnou transformaci, která minimalizuje vzdálenost mezi těmito body. Tato vzdálenost se obvykle označuje jako *zarovnávací chyba*. Po provedení zvolené transformace se opět určí zarovnávací chyba mezi korespondujícími body a postup se iteračně opakuje, dokud není chyba v rámci předem určené tolerance. Přehled soudobých metod registrace mračen bodů je uveden v [49].



(a) (b) (c)

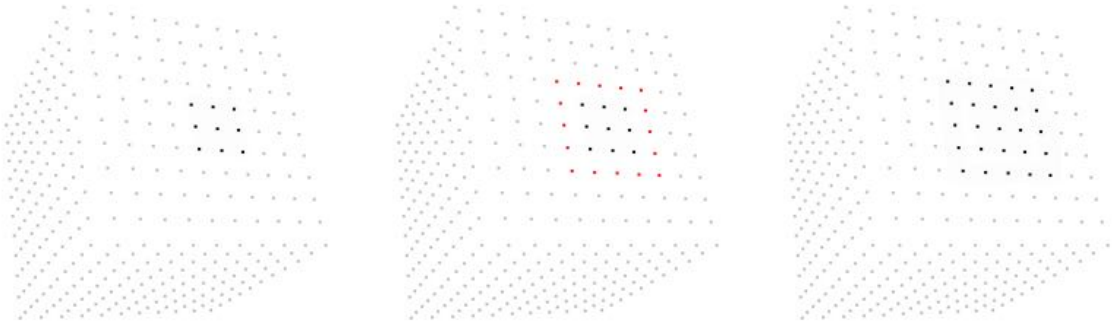
Obrázek 3.12: Registrace mračna bodů Stanford bunny (a) mračno hlavy, (b) mračno těla, (c) složené mračno.

Segmentace

Segmentace je proces rozdělení mračna bodů na tzv. *clustery* (shluky), které jsou tvořeny body se stejnými vlastnostmi. Těmito vlastnostmi mohou být například normálové vektory, křivosti, barvy, souřadnice a další. Přičemž nejčastější aplikace je právě segmentace v závislosti na normále či křivosti. Současné používané segmentační algoritmy jsou uvedeny v [42].

Algoritmy segmentace pracují na jednoduchém principu *region growing* (rostoucí oblast). Jde o postup, ve kterém se nejdříve zvolí prvotní množina bodů a iteračně se k ní přidávají další členy podle zadaného kritéria. Před zahájením samotného procesu se určí klíčové parametry, například odchylka normál, práh křivosti aj. Následně se vybere výchozí bod (*seed point*, obr. 3.14(a)), který je prvotním clusterem, a nalezne se k-okolních bodů (obr. 3.14(b)). Okolní body se testují danými parametry, a buď se do aktuálního shluku přidají, nebo ne. Takto vzniklý cluster se v dalším kroku použije jako výchozí seed point (obr. 3.14(c)), nalezne se k-nejbližších bodů a postup se opakuje, dokud se nepřidají všechny přípustné okolní body. Jakmile je cluster nalezen (žádné další okolní body již nelze přidat), přistoupí k dalšímu libovolnému bodu mimo tento shluk a postup se opakuje, dokud se nezpracují všechny body. Výsledkem segmentace je tedy množina

clusterů obsahujících body s vlastnostmi v dané toleranci. Jak již bylo řečeno, může se například jednat o body s podobnými normálami, tedy body ploch.



(a) (b) (c)

Obrázek 3.13: Region growing segmentace (a) seed point, (b) k-nejbližší okolní body (červené), (c) nový seed point.

Segmentační algoritmus však může některé body vynechat, a tudíž nebudou patřit žádnému clusteru. Tyto body jsou většinou způsobené šumem, a proto je vhodné vstupní data nejdříve filtrovat. Vynechané body se mohou nacházet na pomezí více shluků a při výpočtu nepadnou ani do jednoho z nich (viz obr. 3.14). Této skutečnosti využiji při detekování bodů ostrých hran v kap. 4.3 (str. 66).

(a) (b)

Obrázek 3.14: (a) vstupní data, (b) vysegmentované stěny krychle (šedé), nevybrané body (černé).

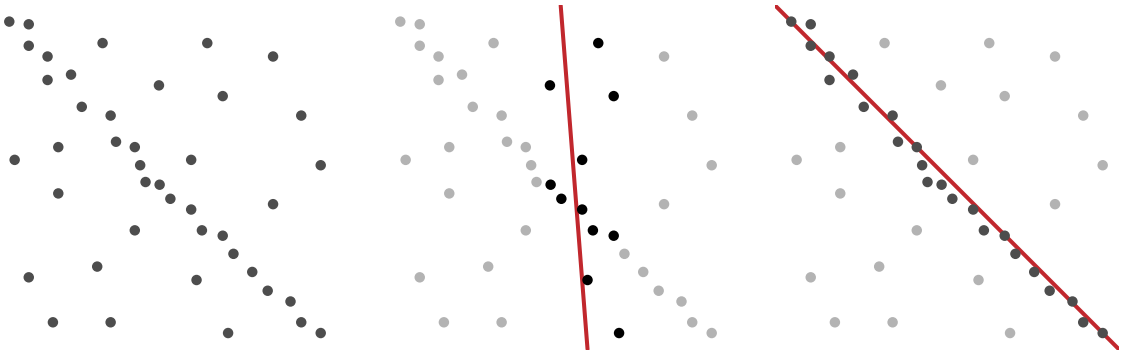
RANSAC

RANSAC (*Random SAmple Consensus*) [20] je metoda zjišťování parametrů matematických modelů v množině pozorovaných dat. Pozorovaná data mohou obsahovat hodnoty, které jsou zcela mimo očekávání. V angličtině se označují jako *outliers* a výpočet parametrů modelu nijak neovlivní. Outliers jsou většinou způsobeny chybou měření a mohou se také projevovat jako šum v získaných datech. RANSAC proto tyto hodnoty outliers při určování parametrů modelu ignoruje a počítá pouze s těmi, které jsou v požadované toleranci.

Algoritmus se nejčastěji používá pro určení parametrů přímek, křivek či rovin, ale své využití má i v oblasti rozpoznávání objektů a tvarů ve vstupních datech. Například pro vyhledání určitého tvaru v obraze. Vyhledávání se provádí iteračně ve dvou základních

krocích: hypotéza a ověření. V prvním kroku se vybere náhodná množina bodů, která se označuje jako tzv. *dataset*, a spočítají se parametry zvoleného modelu pro tuto množinu. Ve druhém kroku se prochází všechny zbývající body a ověřuje se, zda spadají v určené maximální toleranci do nalezeného modelu, či ne. Body, které leží v dané toleranci vzhledem k vypočtenému modelu, se nazývají *inliers*. V tomto kroku se také ověřuje, zda je vypočtený model správný. Ověření se provede porovnáním počtu inliers bodů s přednastavenou hodnotou. Jestliže model nepokryl požadovaný počet bodů, model je považován za chybný a přestoupí se opět k prvním kroku, tedy výběru nové počáteční množiny. Ta se zvolí jiná než předchozí, aby se nezískal stejný chybný model. Body ležící mimo toleranci jsou již zmíněné *outliers*. Množina inliers se následně přidá k původnímu datasetu a vytvoří množinou zvanou *sample consensus* (*vzorový konsenzus*, či *vzorová shoda*). Tato množina se posléze zvolí jako nový dataset a celý proces se opakuje. Algoritmus se ukončí buď v případě, že se nalezeným modelem pokryje určený počet bodů, nebo pokud algoritmus projde určené množství iterací (podle toho, co nastane dříve).

Matematické modely se volí podle dané situace. Vhodnost modelu se jednoduše pozná podle toho, zda se algoritmem vyhledá co největší počet bodů. Pro ilustraci algoritmu uvedeme příklad proložení přímky danými body (viz obr. 3.15).



(a) (b) (c)

Obrázek 3.15: (a) vstupní data, (b) chybný model, (c) správný model s vyznačenými inliers i outliers body.

Triangulace

Triangulace je nejjednodušší a nejrychlejší metodou rekonstrukce povrchu objektu z mračna bodů. Hlavní myšlenkou triangulace je rozdělení mračna bodů do trojic bodů (nacházejících se blízko u sebe) a tedy vytvoření trojúhelníkové sítě tělesa.

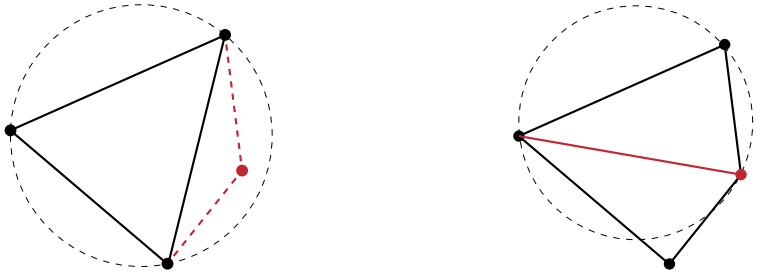
Často používanou metodou je *Delaunayova triangulace*, pojmenovaná podle ruského matematika Borise Nikolajeva Delaunaye, který se tímto tématem zabýval. Tato triangulační metoda se řídí jedním pravidlem: žádný bod neleží uvnitř kružnice opsané libovolnému trojúhelníku. Správné trojúhelníky se označují *Delaunay* a chybné *non-Delaunay*. Tuto podmínku lze jednoduše znázornit na příkladu triangulace 4 bodů v rovině (obr. 3.16).

Matematicky lze toto pravidlo také popsat pomocí součtu protilehlých vnitřních úhlů tohoto 4-úhelníku. Pro čtyři koncyklické body (ležící na jedné kružnic) je součet protilehlých úhlů vždy 180°. Tato situace je tedy mezní polohou bodů mezi Delaunay a non-Delaunay trojúhelníky.

(a) (b)

Obrázek 3.16: (a) Delaunay triangulace, (b) non–Delaunay triangulace.

Algoritmus pracuje následovně. Nejprve se vytvoří prvotní trojúhelník (většinou se jedná o největší možný, který ohraničí téměř všechny ostatní body). Pak se náhodně vybere další bod a vytvoří se nový trojúhelník (obr. 3.17(a)). V dalším kroku se ověří, zda jsou takto vzniklé trojúhelníky správné. Pokud ne, tak se provede záměna strany na druhou možnost tzv. *edge flip* – prohození strany, jak je uvedeno na obr. 3.17.

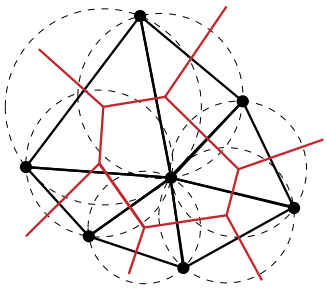


(a) (b)

Obrázek 3.17: (a) přidaný bod s novým trojúhelníkem (červeně) – chybné, (b) zaměněná strana trojúhelníka (červeně).

Proces tedy iterativně přidává nové body, vytváří nové trojúhelníky a ověřuje jejich správnost. V případě chybného trojúhelníku zamění stranu a pokračuje dále, dokud nejsou zpracovány všechny zbylé body.

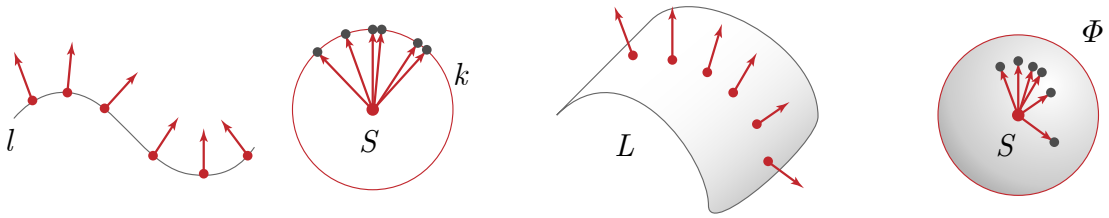
Zajímavostí Delaunay triangulace je její vztah k Voronoiovým diagramům. Tyto diagramy vytváří body, které jsou vždy ve stejné vzdálenosti od zadaných bodů. V případě tří nekolineárních bodů tvořící trojúhelník jsou Voroniovým diagramem osy jeho stran, které prochází středem kružnice jemu opsané. Z konstrukce Delaunayovy triangulace a Voronoiových diagramů je zřejmé, že se jedná o vzájemně inverzní postupy (viz obr. 3.18). Je také vidět, že strana trojúhelníku mezi dvěma body a jejich Voronoiův graf jsou navzájem kolmé.



Obrázek 3.18: Voronoiův diagram (červené) a Delaunayova triangulace (černé).

Gaussovo zobrazení

Body v mračnu bodů můžeme rozdělit do tří kategorií:(i) rovinné body; (ii) body hrany; (iii) rohové body. Tuto klasifikaci je možné provést pomocí *Gaussova zobrazení* (neboli *kruhového obrazu*) [6]. Toto zobrazení promítá bod křivky (či plochy) na kružnici (či kulovou plochu) z jejího středu ve směru příslušného normálového vektoru tohoto bodu (obr 3.19) a je definováno dle předpisu (3.1).



(a) (b)

Obrázek 3.19: Gaussovo zobrazení: (a) body křivky, (b) body plochy.

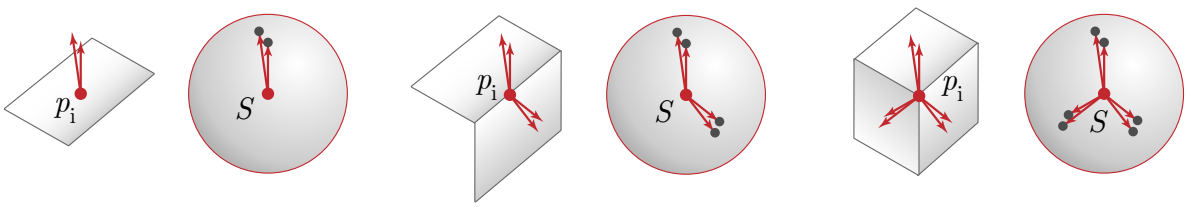
Definice 3.2. Nechť je dána křivka $l \subset \mathbb{R}^2$ a jednotková kružnice $k \subset \mathbb{R}^2$ se středem v libovolném pevném bodě $S = [s_1, s_2] \in \mathbb{R}^2$, pak zobrazení $G : l \to k$ definované předpisem

$$\forall p \in l : G(p) = S + \vec{n_p}, \tag{3.1}$$

kde $\vec{n}_p$ je normálový vektor křivky l v bodě p, se nazývá *Gaussovo zobrazení* křivky l na jednotkovou kružnici k.

Je zřejmé, že množina Gaussových obrazů všech bodů křivky l je část jednotkové kružnice k se středem v bodě S (obr. 3.19(a)). V případě, že p je bodem plochy $L \in \mathbb{R}^3$, pak množina Gaussových obrazů bodů plochy L je část povrchu jednotkové kulové plochy Φ se středem v bodě S (obr. 3.19(b)). Z obr. 3.19 je také zřejmé, že pokud má křivka (plocha) v daném bodě podobné normálové vektory, tak její Gaussův obraz vytvoří na kulové ploše shluky (clustery). Tuto vlastnost využijeme pro klasifikaci daného bodu.

Uvažujme náhodně zvolený bod p_i a normálové vektory plochy v jeho k-nejbližších sousedních bodech. Provedeme Gaussovo zobrazení bodu p_i ve směrech těchto normálových vektorů. Tím vzniknou shluky bodů na kulové ploše, jejichž počet pak určuje typ daného bodu: (1) rovinný bod; (2) bod hrany; (3) bod rohu (obr. 3.20).



(a) (b) (c)

Obrázek 3.20: Gaussovo zobrazení s clustry: (a) rovina, (b) hrana, (c) roh.

3.2. Point cloud library a vizualizační metody

Následující část textu popisuje rozsáhlou softwarovou knihovnu pro zpracování mračen bodů včetně jejich vizualizace.

3.2.1. Point Cloud Library

Softwarová knihovna *PCL* (*Point Cloud Library*) (Rusu a Cousins [53], 2011) je ohromný *open-source, cross-platform* projekt skýtající nepřeberné množství optimalizovaných algoritmů a funkcí pro zpracování a vizualizaci mračen bodů. Vývoj knihovny PCL, jako samostatného projektu věnovaného výhradně práci s mračny bodů, začal v březnu roku 2011. Prvotní verze projektu, na které pracovali výzkumné týmy celého světa, byla dokončena v prosinci téhož roku a zpřístupněna na serveru `pointclouds.org`, kde sídlí dodnes. PCL se stala prestižním prostředkem pro zpracování mračen bodů a na jejím vývoji se začaly podílet i známé korporace jako například Google, Toyota, Leica či nVidia. V dnešní době je ve verzi 1.8.1 a dokumentace je dostupná v [17].
Knihovna obsahuje C++ algoritmy pro tyto hlavní oblasti zpracování bodů:

- filtrace,
- segmentace,
- registrace,
- import a export dat z 3D skenerů,
- rozpoznávání objektů,
- rekonstrukce objektů,
- vizualizace,

které jsou ve velké míře doplněny o užitečné návody. Díky otevřenému zdrojovému kódu a dostupnosti pro dnešní počítačové platformy se na jejím vývoji může podílet prakticky kdokoliv, kdo má zájem o tuto problematiku.

3.2.2. Vizualizace pomocí PCL

Vizualizace je způsob grafického znázornění, zobrazení, pomocí výpočetní techniky. Je nedílnou součástí velkého množství odvětví lidské činnosti včetně strojního inženýrství. V oblasti detekce specifických v mračnu bodů je pod pojmem *vizualizace* také myšleno, zobrazení nalezených bodů prostřednictvím matematického modelu, tedy křivek. Zde se řeší problém proložení křivky nalezenými body, tedy *curve fitting*, k čemuž může být využito matematické regrese. Tato problematika bude podrobně popsána v kap. 3.4 (str. 45).

Visualization ToolKit

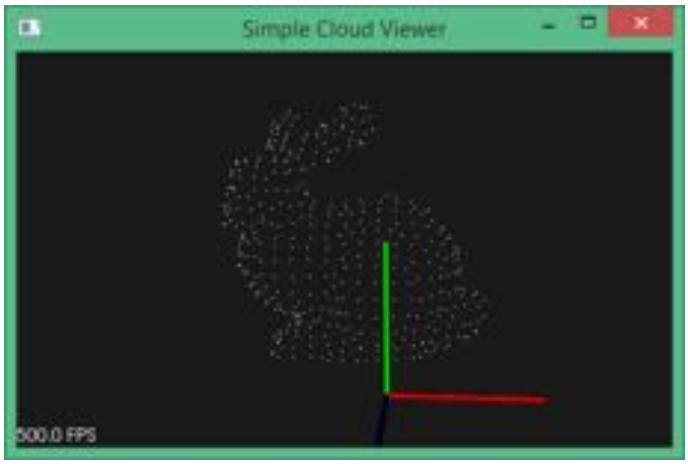
V oblasti zpracovávání mračen bodů se v kombinaci s knihovnou PCL [53] využívá vizualizační knihovna *VTK* (*Visualization ToolKit*) [54]. Jedná se o open-source knihovnu funkcí, která je vyvíjena od roku 1993, představena v roce 1996, a v současné době se používá verze 7.1. O vývoj této knihovny se zasluhuje softwarová firma Kitware, která mimo jiné vyvíjí i CMake.

Kromě vizualizace, VTK obsahuje i modelovací algoritmy, dokáže vytvořit 2D grafy, nebo zpracovávat digitální obrazy. Primárně je však určena k zobrazování a vytváření plošných a objemových vědeckých dat. V této kapitole jsou uvedeny nejpoužívanější implementace knihovny VTK a PCL pro zobrazování zpracovaných dat mračen bodů.

Knihovna PCL má dva základní grafické zobrazovače mračen bodů založené na VTK, a to **CloudViewer** a **PCLVisualizer**.

CloudViewer

CloudViewer je třída knihovny PCL určená k jednoduchému a rychlému zobrazení vloženého mračna bodů (viz obr. 3.21). V zobrazovači je po jeho spuštění možné obraz otáčet, posunovat, přibližovat či oddalovat. Umí zobrazit více mračen v jednom okně a vykresluje také počátek soustavy souřadné včetně vektorů os.

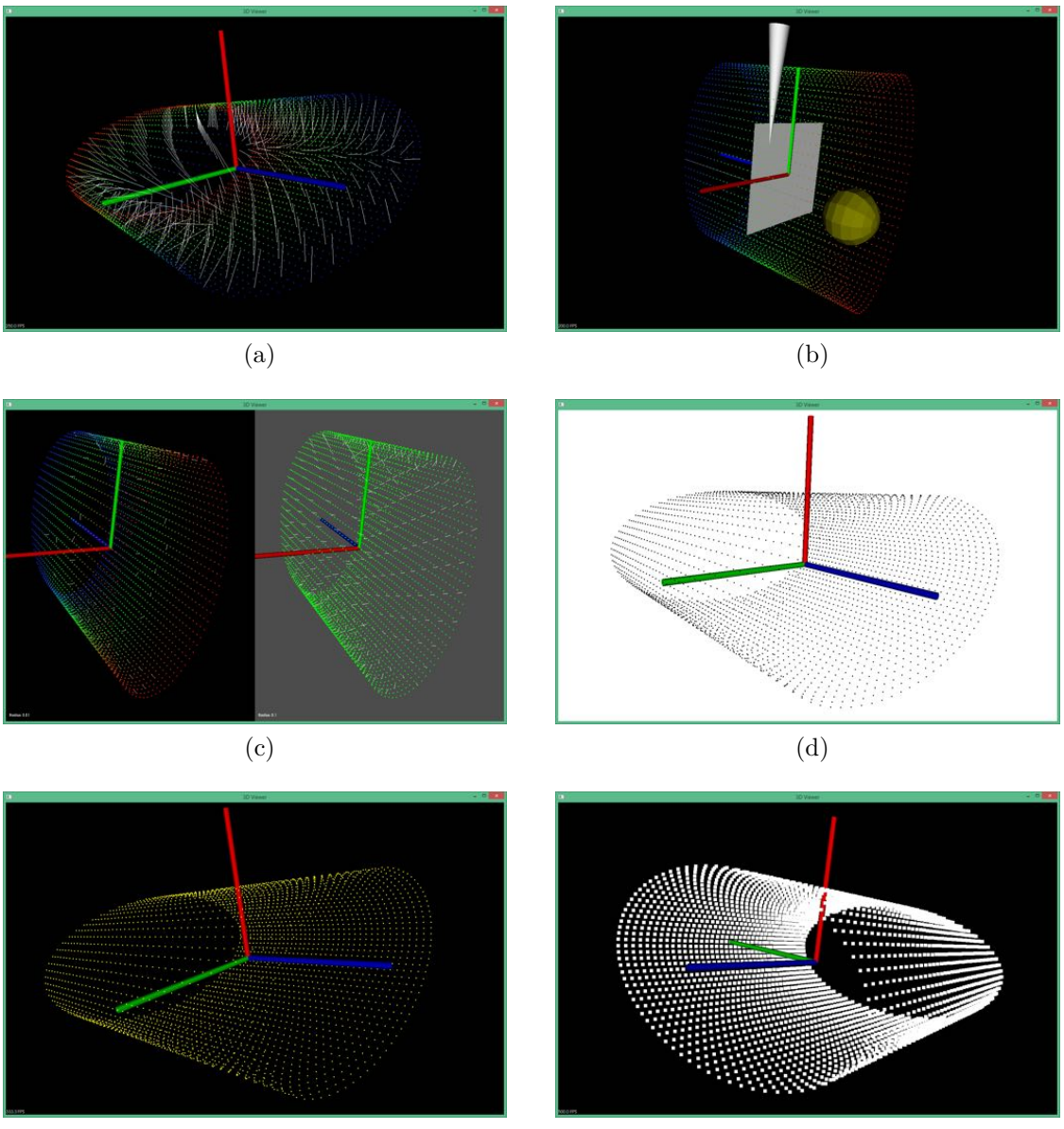


Obrázek 3.21: Výstup z CloudVieweru.

PCLVisualizer

PCLVisualizer oproti předchozí třídě **CloudViewer**, umožňuje pokročilejší možnosti zobrazování mračen bodů a má tyto hlavní přednosti (obr. 3.22):

- zobrazení normál
- přidání elementárních těles, například přímky, kružnice, roviny, koule, válce, kužely, kvádry a další
- přidání více pohledů (*viewportů*)
- grafické úpravy zobrazení (barva pozadí, velikost bodů, barva bodů apod.)
- reakce na tlačítko myši



(a) (b)

(c) (d)

(e) (f)

Obrázek 3.22: Zobrazovací možnosti PCLVisualizeru: (a) normály, (b) tělesa, (c) viewporty, (d) barva pozadí, (e) barva bodů, (f) velikost bodů.

3.3. Křivky používané pro vizualizaci

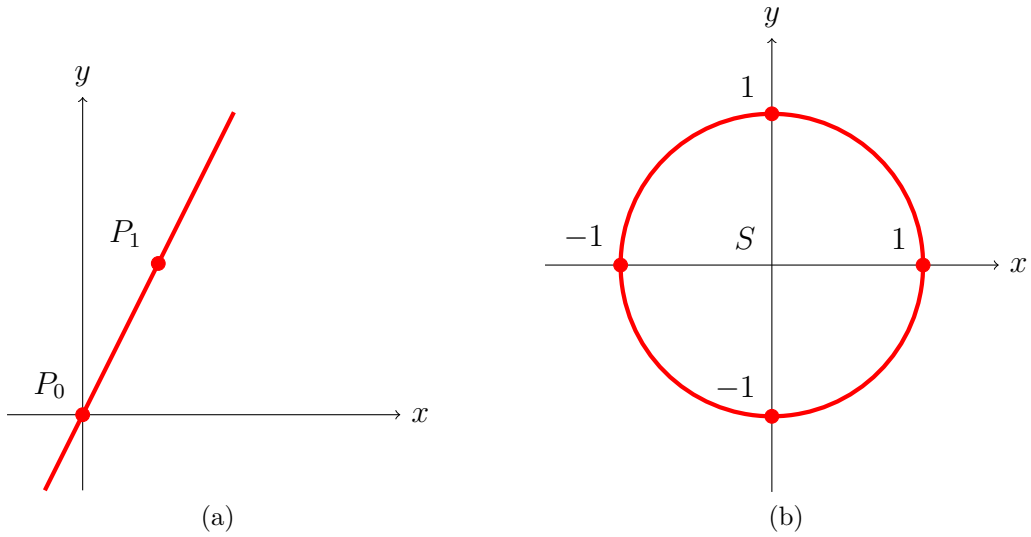
Grafické vykreslení křivky – tedy její *vizualizace* – lze pomocí výpočetní techniky provést mnoha různými způsoby. Stěžejní není samotné vykreslení, ale přijatelný matematický popis, který lze na počítači realizovat. Mezi nejčastěji používané matematické reprezentace křivek patří například: *implicitní* a *parametrická forma*; *polynomiální formy – polynomiální*, *Bézierovy křivky*; či *racionální polynomiální formy – racionální Bézierovy křivky*, *b-spline křivky* nebo *NURBS křivky*. Tyto typy křivek jsou zjednodušeně popsány v této kapitole (více v [45]).

3.3.1. Křivky dané implicitně a parametricky

Implicitně zadaná křivka

Definice 3.3. Křivka $C \in \mathbb{R}^2$ ležící v rovině xy je *implicitně* (také *obecně*) určena rovnicí tvaru $C : f(x, y) = 0$, která udává závislost mezi souřadnicemi x a y bodů P této křivky.

Je tedy snadné ověřit, zda libovolně zvolený bod $M = [x_M, y_M]$ z roviny xy na křivce leží či nikoliv (dosazením jeho souřadnic do vyjádření křivky). Naopak najít bod M tak, aby na křivce ležel, je obtížnější. Navíc je zřejmé, že implicitní zadání křivky je až na násobek konstantou jediné. Implicitní tvar $f(x, y) = 0$ popisuje pouze křivku v rovině xy (záměnou souřadnic lze popsat křivky v rovinách yz i xz), proto tímto způsobem nelze určit prostorové křivky pouze jedinou implicitní rovnicí.



Obrázek 3.23: Grafické znázornění (a) přímky, (b) kružnice.

Příklad 3.4. Přímka procházející body $P_0 = [0, 0]$ a $P_1 = [1, 2]$ má implicitní rovnici $2x - y = 0$ (obr. 3.23(a)). Tatáž přímka má však i rovnici $16x - 8y = 0$, nebo $x - \frac{y}{2} = 0$, či také $-2x + y = 0$ apod. Všechny tyto další rovnice, jak již bylo zmíněno, jsou navzájem ekvivalentní.

Příklad 3.5. Kružnice se středem v počátku a jednotkovým poloměr je dána rovnicí $x^2 + y^2 - 1 = 0$ (obr. 3.23(b)). Pro zobrazení grafu této implicitní funkce je třeba ji vyjádřit ve tvaru $y = y(x)$, tedy $y = \pm\sqrt{1 - x^2}$.

Parametricky zadaná křivka

Definice 3.6. Křivka $C \in \mathbb{R}^3$ je *parametricky* dána po souřadnicích, tedy $C(t) = (x(t), y(t), z(t)), t \in \langle a, b \rangle,\ a, b, t \in \mathbb{R}$, kde implicitní funkce $x(t)$, $y(t)$ a $z(t)$ popisují chování jednotlivých souřadnic bodů této křivky. Vektor $C(t) = (x(t), y(t), z(t))$ je někdy označován jako *parametrizace* křivky.

Parametrické vyjádření křivky také není jediné. Libovolná křivka může mít mnoho parametrizací v závislosti na volbě jednotlivých souřadnicových funkcí.

Ověřit, zda libovolný bod $M = [x_M, y_M, z_M]$ na křivce leží či nikoliv, již není tak snadné jako v případě implicitního tvaru. Zde je třeba najít takovou hodnotu parametru t, aby byly splněny všechny souřadnicové funkce současně. Bod M na křivce leží v případě, že takový parametr existuje, v opačném případě ne. Naopak určit souřadnice bodu M tak, aby na křivce ležel, je snadné (přímo z definice křivky).

Příklad 3.7. Přímka určená body P_0 a P_1 z předchozí ilustrace 3.23(a) má například parametrický tvar:

$$C(t) = (t, 2t),\ t \in \mathbb{R} \tag{3.2}$$

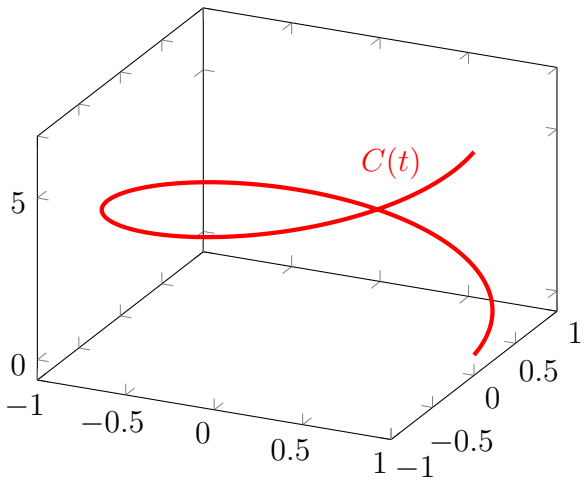
Příklad 3.8. Kružnice se středem v počátku a poloměrem 1 (obr. 3.23(b)) má například tuto parametrizaci:

$$C(t) = (\cos(t), \sin(t))),\ t \in \langle 0, 2\pi \rangle. \tag{3.3}$$

Parametrické vyjádření křivky představuje posun bodů po křivce v závislosti na změně parametru t. Rychlost tohoto pohybu je dán první derivací vektoru parametrizace $C(t)$ podle proměnné t, zrychlení pak druhou derivací podle stejné proměnné.

Příklad 3.9. Šroubovice s osou z, poloměrem 1 a $v_0 = 1$ (obr. 3.24) má touto parametrizaci:

$$C(t) = (\cos(t), \sin(t), t),\quad t \in \langle 0, 2\pi \rangle. \tag{3.4}$$



Obrázek 3.24: Šroubovice.

3.3.2. Obecné polynomiální křivky

Pro reprezentaci křivek ve výpočetní technice je vhodné použít funkce, které jsou snadno implementovatelné, přesně zpracovatelné a matematicky dobře popsatelné. Často používané funkce jsou polynomy a pomocí nich jsou určeny například *obecné polynomiální* či *Bézierovy křivky*.

Definice 3.10. Obecnou polynomiální křivku $C \in \mathbb{R}^3$ n-tého stupně určíme polynomem téhož stupně následovně:

$$C(t) = (x(t), y(t), z(t)) = \sum_{i=0}^{n} a_i t^i, \ t \in \langle a, b \rangle, \tag{3.5}$$

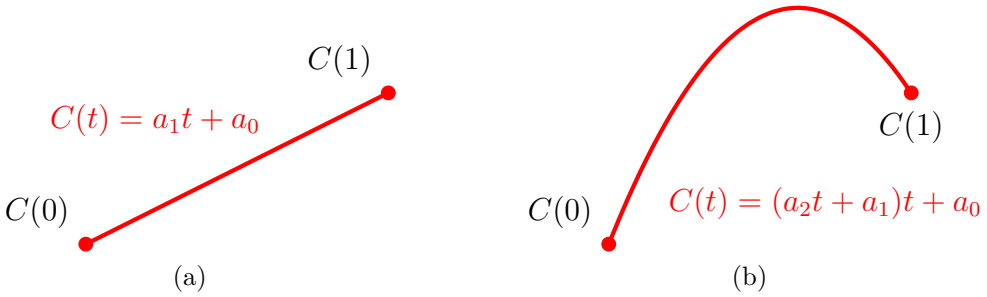
kde $a_i = (x_i, y_i, z_i)$ jsou *koeficienty* a funkce t^i jsou *bázové funkce* této křivky na intervalu $\langle a, b \rangle$. Souřadnicové funkce jsou dány polynomy:

$$x(t) = \sum_{i=0}^{n} x_i t^i; \ y(t) = \sum_{i=0}^{n} y_i t^i; \ z(t) = \sum_{i=0}^{n} z_i t^i. \tag{3.6}$$

Příklad 3.11. Pro $n = 1$ získáme úsečku mezi body $C(0)$ a $C(1)$ tvaru $C(t) = \sum_{n=0}^{1} a_i t^i = a_1 t + a_0, \ t \in \langle 0, 1 \rangle$ (obr. 3.25(a)).

Příklad 3.12. Pro $n = 2$ získáme část paraboly mezi body $C(0)$ a $C(1)$ ve tvaru $C(t) = \sum_{n=0}^{2} a_i t^i = a_2 t^2 + a_1 t + a_0 = (a_2 t + a_1)t + a_0, \ t \in \langle 0, 1 \rangle$ (obr. 3.25(b)).

Příklad 3.13. Pro $n = 3$ získáme obecnou křivku danou rovnicí $C(t) = \sum_{n=0}^{3} a_i t^i = a_3 t^2 + a_2 t^2 + a_1 t + a_0 = ((a_3 t + a_2)t + a_1 t)t + a_0$.



Obrázek 3.25: Znázornění křivek stupně (a) n=1, (b) n=2.

Z předchozích příkladů je patrné, že bod na křivce lze v pevně dané hodnotě parametru $t_0 \in \langle 0, 1 \rangle$ určit pomocí Hornerova schematu [45].

$$\begin{aligned}
n = 1: &\quad C(t_0) = a_1 t_0 + a_0 \\
n = 2: &\quad C(t_0) = (a_2 t_0 + a_1)t_0 + a_0 \\
n = 3: &\quad C(t_0) = ((a_3 t_0 + a_2)t_0 + a_1)t_0 + a_0 \\
&\quad \vdots \\
n = n &\quad C(t_0) = (\ldots(a_n t_0 + a_{n-1})t_0 + a_{n-2})t_0 + \cdots + a_0
\end{aligned}$$

Algoritmus 1 Horner

```
1: procedure HORNER(a, t, t0, C)          ▷ Algoritmus vypočítá bod na křivce C
2:     C = a[n]
3:     for i = n − 1; i >= 0; i − − do
4:         C = C * t0 + a[i]
5:     end for
6:     return C
7: end procedure
```

3.3.3. Bézierovy křivky

Speciálním typem polynomiálních křivek jsou tzv. *Bézierovy křivky* [8], [29], [21], [23] popsané následující definicí.

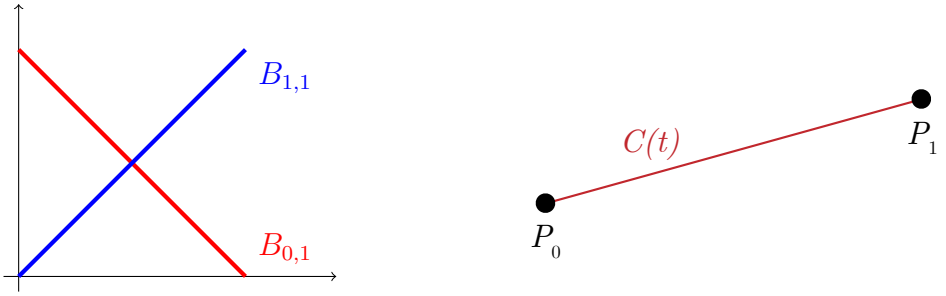
Definice 3.14. *Bézierova křivka* $C \in \mathbb{R}^2$ n-tého stupně je určena předpisem

$$C(t) = \sum_{i=0}^{n} B_{i,n}(t) P_i, \ t \in \mathbb{R}, \ t \in \langle 0, 1 \rangle, \tag{3.7}$$

kde P_i jsou *řídicí body* (tvoří *řídicí polygon*) a $B_{i,n}(t)$ jsou *bázové funkce* n-tého stupně definované pomocí *Bernsteinových polynomů* téhož stupně ve tvaru

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i} = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}. \tag{3.8}$$

Příklad 3.15. Pro hodnotu $n = 1$ získáme lineární Bézierovu křivku (obr. 3.26), která má bázové funkce $B_{0,1}(t) = 1 - t$ a $B_{1,1}(t) = t$ a rovnice (3.7) má tvar $C(t) = P_0(1-t) + P_1 t$, jde tedy o úsečku mezi body P_0 a P_1.

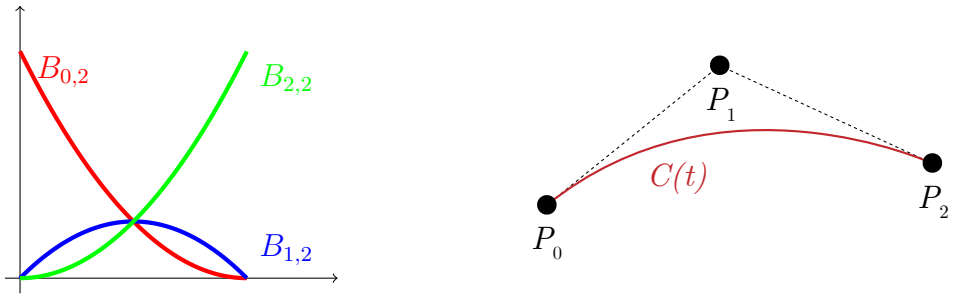


(a) (b)

Obrázek 3.26: Lineární Bézierova křivka (a) bázové funkce, (b) křivka.

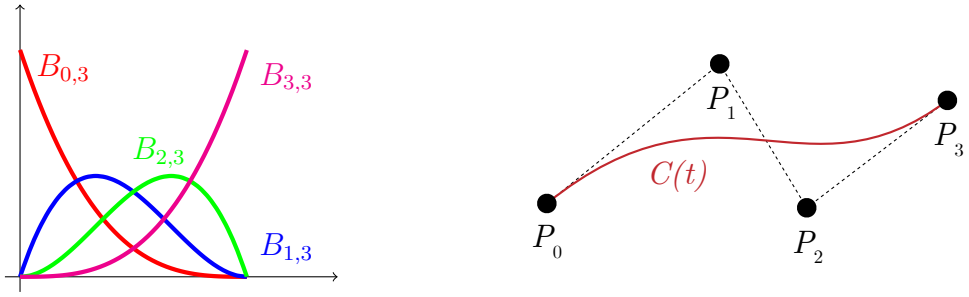
Příklad 3.16. Pro hodnotu $n = 2$ získáme kvadratickou Bézierovu křivku (obr. 3.27). Bázové funkce jsou $B_{0,2}(t) = (1-t)^2$, $B_{1,2}(t) = 2t(1-t)$ a $B_{2,2}(t) = t^2$. Rovnice křivky (3.7) je tvaru $C(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2$. Jde o část paraboly mezi body P_0 a P_2.

Příklad 3.17. Pro hodnotu $n = 3$ získáme kubickou Bézierovu křivku (obr. 3.28), jejíž bázové funkce jsou $B_{0,3}(t) = (1-t)^3$, $B_{1,3}(t) = 3t(1-t)^2$, $B_{2,3}(t) = 3t^2(1-t)$ a $B_{3,3}(t) = t^3$. Rovnice (3.7) má tvar $C(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_2 + 3t^2(1-t)P_1 + t^3 P_3$.



(a) (b)

Obrázek 3.27: Kvadratická Bézierova křivka (a) bázové funkce, (b) křivka.



(a) (b)

Obrázek 3.28: Kubická Bézierova křivka (a) bázové funkce, (b) křivka.

Věta 3.18. Bázové funkce $B_{i,n}(t)$ mají tyto klíčové vlastnosti

- $\forall i, n: \ B_{i,n}(t) \geq 0, \ t \in \langle 0, 1 \rangle$,
- $\sum_{i=0}^{n} B_{i,n}(t) = 1, \ t \in \langle 0, 1 \rangle$,
- $B_{0,0}(0) = B_{n,n}(1) = 1$.

Důkaz lze najít v [45].

Příklad 3.19. Uvažujme znovu $n = 2$, kdy dostáváme rovnici kvadratické Bézierovy křivky

$$C(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2. \tag{3.9}$$

Rovnici můžeme následovně přepsat

$$C(t) = (1-t)((1-t)P_0 + tP_1)) + t((1-t)P_1 + tP_2). \tag{3.10}$$

Je zřejmé, že $(1-t)P_0 + tP_1$ a $(1-t)P_1 + tP_2$ jsou lineární Bézierovy křivky po řadě mezi body P_0, P_1 a P_1, P_2. Tedy kvadratická Bézierova křivka vzniká jako interpolace dvou lineárních křivek. Uvažujeme-li pevné $t_0 \in \langle 0, 1 \rangle$ a označíme-li $P_{1,0} = (1-t_0)P_0 + t_0 P_1$, $P_{1,1} = (1-t_0)P_1 + t_0 P_2$ a $P_{2,0} = (1-t_0)P_{1,0} + t_0 P_{1,1}$, pak platí, že $C(t_0) = P_{2,0}$. Při volbě $t_0 = 0,5$ získáme bod křivky, jak je uvedeno na obr. 3.29(a). Stejným způsobem je zkonstruován bod kubické Bézierovy křivky, jehož konstrukce je uvedena na obr. 3.29(b).

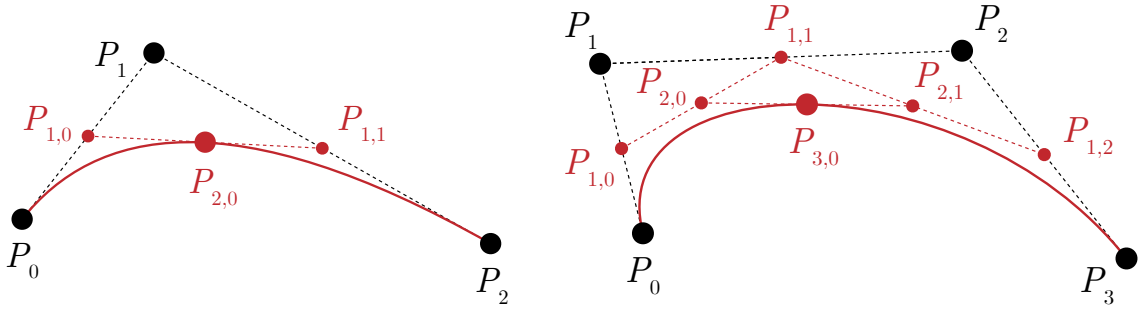
Analogicky lze postupovat pro křivky vyšších stupňů. Označíme-li Bézierovu křivku n-tého stupně jako $C_n(P_0, \ldots, P_n)$ získáme tento předpis

$$C_n(P_0, \ldots, P_n) = (1-t)C_{n-1}(P_0, \ldots, P_{n-1}) + tC_{n-1}(P_1, \ldots, P_n). \tag{3.11}$$

Přeznačením P_i na $P_{0,i}$ a pevném $t_0 \in \langle 0, 1\rangle$ tato rovnice přechází do tvaru $C(t_0) = P_{n,0}(t_0)$ a získáme tím rekurzivní algoritmus pro výpočet bodu křivky v následujícím tvaru:

$$P_{k,i}(t_0) = (1-t_0)P_{k-1,i}(t_0) + t_0 P_{k-1,i+1}(t_0), \text{ pro } \begin{cases} k & = 1, 2, \ldots, n \\ i & = 0, 1, \ldots, n-k. \end{cases} \tag{3.12}$$

Tento rekurzivní předpis se nazývá *deCasteljauův algoritmus* a jeho průběh je znázorněn na obr. 3.29 a jeho pseudokód je uveden v Algoritmu 2.



(a) (b)

Obrázek 3.29: deCasteljauův algoritmus v bodě $t_0 = 0,5$ (a) kvadratické, (b) kubické Bézierovy křivky.

Algoritmus 2 deCasteljau

```
1:  procedure DECASTELJAU(P, n, t0, C)        ▷ Algoritmus vypočítá bod na křivce C
2:      for (i = 1; i >= n; i + +) do
3:          Q[i] = P[i]                        ▷ zachování vstupních řídicích bodů P
4:      end for
5:      for (k = 1; k <= n; k + +) do
6:          for (i = 0; i <= n – k; i + +) do
7:              Q[i] = (1 – t0) * Q[i] + t0 * Q[i + 1]
8:          end for
9:      end for
10:     C = Q[0]
11:     return C
12: end procedure
```

Oproti polynomiálním křivkám lze tvar Bézierovy křivky jednoduše změnit upravením polohy jednotlivých řídicích bodů a její konstrukce využitím deCasteljauva algoritmu je méně citlivá na zaokrouhlovací chybu než Hornerovo schema z případu obecných polynomiálních křivek. Řadu křivek není možné určit pomocí polynomu. Jedná se například o kuželosečky. Tyto křivky však lze popsat racionálními funkcemi, a proto byla zavedena racionální varianta Bézierových křivek [19].

3.3.4. Racionální Bézierovy křivky

Jak již bylo zmíněno, polynomiální funkce nejsou schopné popsat kuželosečkové oblouky. Tyto oblouky lze popsat pomocí dvojice polynomů ve tvaru zlomku, tedy ve formě racionální funkce následovně

$$x(t) = \frac{X(t)}{W(t)},\ y(t) = \frac{Y(t)}{W(t)},\ t \in I \tag{3.13}$$

kde $X(t)$, $Y(t)$ a $W(t)$ jsou polynomy a I je interval. Je také vidět, že obě souřadnicové funkce mají stejný jmenovatel.

Příklad 3.20. Mějme kružnici s jednotkovým poloměrem umístěnou v počátku. Její zadání v racionálním tvaru je například

$$C(t) = (x(t), y(t)),\ \text{kde } x(t) = \frac{1-t^2}{1+t^2},\ y(t) = \frac{2t}{1+t^2},\ t \in \mathbb{R}. \tag{3.14}$$

Příklad 3.21. Elipsa s umístěnou v počátku s hlavní osou y, vedlejší x, hlavní poloosou rovnu 2 a vedlejší 1. Racionálním tvar je například

$$C(t) = (x(t), y(t)),\ \text{kde } x(t) = \frac{1-t^2}{1+t^2},\ y(t) = \frac{4t}{1+t^2},\ t \in \mathbb{R}. \tag{3.15}$$

I v těchto případech si můžeme všimnout, že jmenovatel souřadnicových funkcí je stejný.

Definice 3.22. *Racionální Bézierova křivka* $C \in \mathbb{R}^2$ n-tého stupně je určena předpisem

$$C(t) = \frac{\sum_{i=0}^{n} B_{i,n}(t) w_i P_i}{\sum_{i=0}^{n} B_{i,n}(t) w_i},\ t \in \langle 0, 1\rangle, \tag{3.16}$$

kde P_i, $R_{i,n}(t)$ jsou stejné pojmy jako v předchozí části textu a $w_i \in \mathbb{R}$ jsou tzv. *váhy*.

Váhy $w_i \in \mathbb{B}$ budeme v další části uvažovat pouze kladné, tedy $w_i > 0$. Definici můžeme přepsat do tvaru

$$C(t) = \sum_{i=0}^{n} R_{i,n}(t) P_i,\ t \in \langle 0, 1\rangle,\ \text{kde } R_{i,n}(u) = \frac{B_{i,n}(t) w_i}{\sum_{j=0}^{n} B_{j,n}(t) w_j}, \tag{3.17}$$

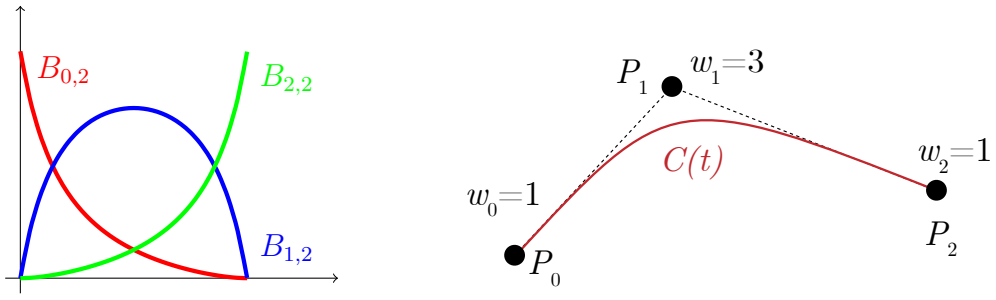
přičemž $R_{i,n}$ jsou tzv. *racionální bázové funkce*.

Příklad 3.23. Mějme $n = 2$ a váhy $w_0 = 1$, $w_1 = 3$ a $w_2 = 1$. Bázové funkce $B_{j,2}$ jsou tvaru $B_{0,2}(t) = (1-t)^2$, $B_{1,2}(t) = 2t(1-t)$ a $B_{2,2}(t) = t^2$ a společný jmenovatel racionálních bázových funkcí je tedy tvaru $\sum_{j=0}^{2} B_{j,2}(t) w_j = B_{0,2}(t) w_0 + B_{1,2}(t) w_1 + B_{2,2}(t) w_2$. Po dosazení získáme $1 \cdot (1-t)^2 + 3 \cdot 2t(1-t) + 1 \cdot t^2$ a po úpravě je společný jmenovatel ve tvaru $-4t^2 + 4t + 1$. Po dosazení do rovnice (3.17) získáme tyto racionální bázové funkce

$$R_{0,2}(t) = \frac{(1-t)^2 \cdot 1}{-4t^2 + 4t + 1},\ R_{1,2}(t) = \frac{2t(1-t) \cdot 3}{-4t^2 + 4t + 1},\ R_{2,2}(t) = \frac{t^2 \cdot 1}{-4t^2 + 4t + 1}. \tag{3.18}$$

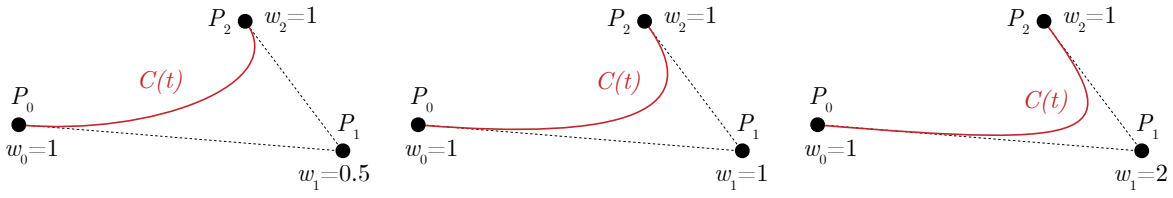
Racionální Bézierova křivka stupně 2 s těmito vahami (obr. 3.30) má tedy rovnici

$$C(t) = \frac{(1-t)^2 \cdot 1}{-4t^2 + 4t + 1} P_0 + \frac{2t(1-t) \cdot 3}{-4t^2 + 4t + 1} P_1 + \frac{t^2 \cdot 1}{-4t^2 + 4t + 1} P_2,\ t \in \langle 0, 1\rangle. \tag{3.19}$$



(a) (b)

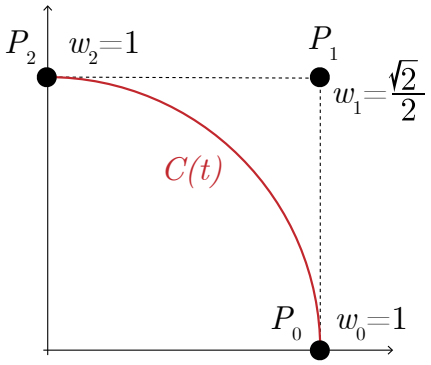
Obrázek 3.30: Kvadratická racionální Bézierova křivka (a) bázové funkce, (b) křivka.



(a) (b) (c)

Obrázek 3.31: Kvadratická racionální Bézierova křivka s různými vahami w_1 (a) $w_1 = 0.5$, (b) $w_1 = 1$, (c) $w_1 = 2$.

Příklad 3.24. Oblouk kružnice se středem v počátku, poloměrem 1 v prvním kvadrantu můžeme pomocí kvadratické racionální Bézierovy křivky určit následovně. Řídicí body zvolíme $P_0 = [1, 0]$, $P_1 = [1, 1]$ a $P_2 = [0, 1]$ a příslušné váhy $w_0 = 1$, $w_1 = \frac{\sqrt{2}}{2}$ a $w_2 = 1$. Výsledná křivka je zobrazena na obr. 3.32.



Obrázek 3.32: Reprezentace části kružnice pomocí kvadratické Bézierovy křivky.

Věta 3.25. Racionální bázové funkce $R_{i,n}(t)$ mají tyto klíčové vlastnosti

- $\forall i, n: \ R_{i,n}(t) \geq 0, \ t \in \langle 0, 1 \rangle$,
- $\sum_{i=0}^{n} R_{i,n}(t) = 1, \ t \in \langle 0, 1 \rangle$,
- $R_{0,0}(0) = R_{n,n}(1) = 1$.

Důkaz viz [45].

Je také patrné, že polynomické Bézierovy křivky (kap. 3.3.3 str. 29) jsou speciálním případem racionálních Bézierových křivek, kdy je jmenovatel racionálních bázových funkcí roven 1, tedy

$$\sum_{i=1}^{n} B_{j,n}(t)w_j = 1, \tag{3.20}$$

čehož lze dosáhnout přiřazením každému řídicímu bodu stejnou váhu rovnu 1, o čemž se můžeme snadno přesvědčit v předchozím příkladě kvadratické racionální Bézierovy křivky, tedy když

$$B_{0,2}(t)w_0 + B_{1,2}(t)w_1 + B_{2,2}(t)w_2 = (1-t)^2 \cdot 1 + 2t(1-t) \cdot 1 + t^2 = 1, \tag{3.21}$$

pak

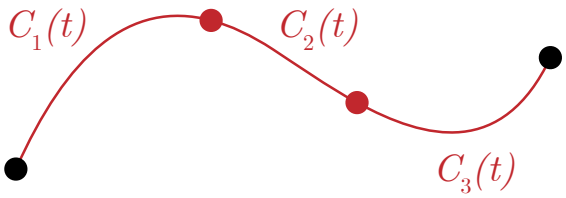
$$C(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2, \ t \in \langle 0, 1 \rangle \tag{3.22}$$

je rovnicí polynomické Bézierovy křivky (viz kap. 3.3.3 str. 29).

Jak je vidět z konstrukce polynomiálních či racionálních Bézierových křivek, s přibývajícím počtem řídicích bodů, narůstá stupeň křivky. Křivky vysokého stupně se ve výpočetní technice obtížněji zpracovávají (vysoké nároky na paměť, numericky nestabilní výpočty) a tudíž nejsou vhodné pro proložení složitějších tvarů. Proto vhodným řešením je tyto křivky rozdělit na po částech polynomiální (viz následující kap. 3.3.5) či racionální křivky (viz kap. 3.3.6 str. 42).

3.3.5. B-spline

B-spline (basis spline) jsou po částech polynomiální křivky, které umožňují počítačovou interpretaci složitějších křivek prokládaných větším počtem bodů. Jednotlivé části na sebe spojitě navazují a v místě přechodu vzniká hladký spoj (viz obr. 3.33). Křivky jsou inspirovány *spline křivkami*, které vytvořil Isaac J. Schoenberg v 50. letech 20. století, a byly představeny mezi lety 1973 a 1974 pány Williamem J. Gordonem a Richardem F. Reinsenfeldem [24], [52].



Obrázek 3.33: Princip b-spline křivek: jednotlivé části $C_1(t), C_2(t), C_3(t)$ propojené v červených bodech.

Definice 3.26. *B-spline křivka* $C \in \mathbb{R}^2$ stupně p je určena předpisem

$$C(u) = \sum_{i=0}^{n} N_{i,p}(u)P_i, \ u \in \langle a, b \rangle, \ a, b \in \mathbb{R}, \tag{3.23}$$

kde P_i jsou řídicí body (určující řídicí polygon) a $N_{i,p}(u)$ jsou *b-spline bázové funkce stupně* p. Tyto bázové funkce jsou definovány rekurzivní formou [13], [14].

$$N_{i,0}(u) = \begin{cases} 1, & u_i \leq u < u_{i+1} \\ 0 & \text{jinde} \end{cases} \tag{3.24}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{3.25}$$

na neklesající posloupnosti reálných čísel – *uzlovém vektoru* U tvaru

$$U = \{u_0, u_1 \ldots, u_m\} \tag{3.26}$$

kde $u_i, i = 0, 1, \ldots, m; m \in \mathbb{Z}$ jsou *uzly*. Počet řídicích bodů $n+1$, stupeň křivky p a počet uzlů $m+1$ je určen vztahem

$$m = n + p + 1. \tag{3.27}$$

Pokud se v uzlovém vektoru opakuje první a poslední hodnota právě $(p+1)$-krát, jedná se o křivku, která prochází prvním a posledním řídicím bodem. Tento tvar uzlového vektoru se v praxi velmi často používá.

$$U = \{\underbrace{a, \ldots, a}_{p+1}, \underbrace{u_{p+1}, \ldots, u_{m-p-1}}_{n-p}, \underbrace{b, \ldots, b}_{p+1}\}, \tag{3.28}$$

Vnitřní uzly uzlového vektoru se mohou opakovat nejvýše p-krát. Při tomto opakování (mluvíme o *násobnosti uzlu*) dochází ke snížení spojitosti v odpovídajícím bodě na C^0. Jestliže se uzel opakuje k-krát ($k \leq p$), pak v tomto uzlu se sníží spojitost na C^{p-k}.
V praxi se také často pracuje s uzlovým vektorem na intervalu $\langle 0, 1\rangle$. Libovolný rozsah uzlového vektoru lze na tento interval převést.

Příklad 3.27. B-spline křivka stupně $p = 2$ určená řídicími body P_0, P_1, P_2 ($n = 2$). Počet m je tedy roven $m = n + p + 1 = 2 + 2 + 1 = 5$ (6 uzlových bodů), a uzlový vektor zvolme $U = \{u_0 = 0, u_1 = 0, u_2 = 0, u_3 = 1, u_4 = 1, u_5 = 1\}$. Bázové b-spline funkce stupně $0, 1,$ a 2 jsou tvaru

p=0

$$N_{0,0} = N_{1,0} = 0, \ -\infty < u < \infty \tag{3.29}$$

$$N_{2,0} = \begin{cases} 1, & u \in \langle 0, 1) \\ 0 & \text{jinde} \end{cases} \tag{3.30}$$

$$N_{3,0} = N_{4,0} = 0, \ -\infty < u < \infty \tag{3.31}$$

p=1

$$N_{0,1}(u) = \frac{u-u_0}{u_1-u_0}N_{0,0} + \frac{u_2-u}{u_2-u_1}N_{1,0} = \frac{u-0}{0-0}N_{0,0} + \frac{0-u}{0-0}N_{1,0} = \begin{cases} (1-u), & u \in \langle 0,1) \\ 0 & \text{jinde} \end{cases} \tag{3.32}$$

$$N_{1,1}(u) = \frac{u-u_1}{u_2-u_1}N_{1,0} + \frac{u_3-u}{u_3-u_2}N_{2,0} = \frac{u-0}{0-0}N_{1,0} + \frac{1-u}{1-0}N_{2,0} = \begin{cases} 1-u, & u \in \langle 0,1) \\ 0 & \text{jinde} \end{cases} \tag{3.33}$$

$$N_{2,1}(u) = \frac{u-u_2}{u_3-u_2}N_{2,0} + \frac{u_4-u}{u_4-u_3}N_{3,0} = \frac{u-0}{1-0}N_{2,0} + \frac{1-u}{1-1}N_{3,0} = \begin{cases} u, & u \in \langle 0,1) \\ 0 & \text{jinde.} \end{cases} \tag{3.34}$$

$$N_{3,1}(u) = \frac{u-u_3}{u_4-u_3}N_{3,0} + \frac{u_5-u}{u_5-u_4}N_{4,0} = \frac{u-1}{1-1}N_{3,0} + \frac{1-u}{1-1}N_{4,0} = 0, \ -\infty < u < \infty \tag{3.35}$$

p=2

$$N_{0,2}(u) = \frac{u-u_0}{u_2-u_0}N_{0,1} + \frac{u_3-u}{u_3-u_1}N_{1,1} = \frac{u-0}{0-0}N_{0,1} + \frac{1-u}{1-0}N_{1,1} = \begin{cases} (1-u)^2, & u \in \langle 0,1) \\ 0 & \text{jinde} \end{cases} \tag{3.36}$$

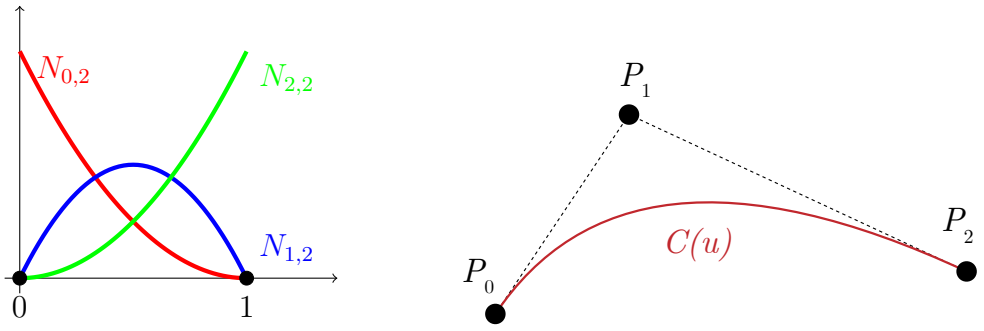
$$N_{1,2}(u) = \frac{u-u_1}{u_3-u_1}N_{1,1} + \frac{u_4-u}{u_4-u_2}N_{2,1} = \frac{u-0}{1-0}N_{1,1} + \frac{1-u}{1-0}N_{2,1} = \begin{cases} 2u(1-u), & u \in \langle 0,1) \\ 0 & \text{jinde} \end{cases} \tag{3.37}$$

$$N_{2,2}(u) = \frac{u-u_2}{u_4-u_2}N_{2,1} + \frac{u_5-u}{u_5-u_3}N_{3,1} = \frac{u-0}{1-0}N_{1,1} + \frac{1-u}{1-1}N_{2,1} = \begin{cases} u^2, & u \in \langle 0,1) \\ 0 & \text{jinde.} \end{cases} \tag{3.38}$$

Kvadratická b-spline křivka má tedy rovnici

$$C(u) = N_{0,2}(u)P_0 + N_{1,2}(u)P_1 + N_{2,2}(u)P_2 = (1-u)^2P_0 + 2u(1-u)P_1 + u^2P_2, u \in \langle 0,1). \tag{3.39}$$

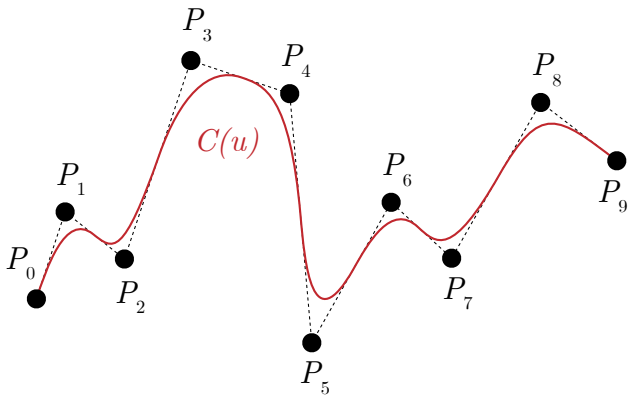
Při této volbě b-spline křivky, kdy $2 = n = p = 2$ a $U = \{0,0,0,1,1,1\}$ získáváme kvadratickou Bézierovu křivku (obr. 3.34). Jinak řečeno, jestliže $n = p$ a $U = \{\underbrace{0,\ldots,0}_{p+1}\underbrace{1,\ldots,1}_{p+1}\}$, pak takto určená b-spline křivka je Bézierovou křivkou téhož stupně.



(a) (b)

Obrázek 3.34: Kvadratická Bézierova křivka popsaná jako kvadratická b-spline křivka s uzlovým vektorem $U = \{0, 0, 0, 0, 1, 1, 1, 1\}$ (a) bázové b-spline funkce, (b) křivka.

Příklad 3.28. Mějme řídicí body $P_0, P_1, \ldots, P_9$, a proložíme jimi b-spline křivku stupně 2 s uzlovým vektorem $U = \{0, 0, 0, 1/8, 2/8, 3/8, 4/8, 5/8, 6/8, 7/8, 1, 1, 1\}$. Grafické znázornění je uvedeno na obr. 3.35.



Obrázek 3.35: B-spline křivka stupně 2 určená zadanými řídicími body.

Věta 3.29. B-spline bázové funkce $N_{i,p}(u)$ mají tyto klíčové vlastnosti

- $\forall i, n: \ N_{i,p}(u) \geq 0, \ u \in \langle 0, 1 \rangle$,
- $\sum_{i=0}^{n} N_{i,p}(u) = 1, \ u \in \langle 0, 1 \rangle$,
- $N_{0,p}(0) = N_{n,p}(1) = 1$.

Důkaz je možné najít v [45].

Je zřejmé, že pro vyjádření bázových b-spline funkcí stupně p je nutné určit všechny bázové b-spline stupňů $0, 1, \ldots, p$. Tyto bázové funkce lze určit pomocí následujících algoritmů nebo pomocí *De Boorova algoritmu* [13], [14], který je zobecněním De Casteljauova algoritmu.

Výpočet b-spline bázových funkcí

Z rekurzivního tvaru (3.24), (3.25) jsou zřejmé tyto vlastnosti b-spline bázových funkcí:

- bázové funkce jsou určeny uzlovým vektorem U a stupněm p b-spline křivky;

- $N_{i,0}(u), i = 0, 1, \ldots, n$ je krokovací funkce, která je rovna nule na celém původním intervalu $\langle 0, 1\rangle$ kromě bodů z intervalu $\langle u_i, u_{i+1}\rangle$;
- $N_{i,p}, i = 0, 1, \ldots, n;\ p > 0$ je lineární kombinací dvou bázových funkcí stupně $(p-1)$;
- v rovnici (3.25) může mít některý koeficient roven ∞, který však definujeme jako 0;
- $N_{i,p}(u)$ jsou reálné po částech polynomické funkce, které jsou vždy omezené na požadovaném intervalu $\langle u_0, u_m\rangle$;
- zprava otevřený interval $\langle u_i, u_{i+1})$ se většinou označuje jako *i-tý uzlový interval* (v angličtině *i-th knot span*);
- výpočet b-spline bázových funkcí $N_{i,p}(u)$ vytváří trojúhelníkové schema ve tvaru

$$\begin{array}{ccccc}
N_{0,0} & & & & \\
 & N_{0,1} & & & \\
N_{1,0} & & N_{0,2} & & \\
 & N_{1,1} & & N_{0,3} & \\
N_{2,0} & & N_{1,2} & & \\
 & N_{2,1} & & N_{1,3} & \\
N_{3,0} & & N_{2,2} & \vdots & \\
 & N_{3,1} & \vdots & & \\
N_{4,0} & \vdots & & & \\
\vdots & & & &
\end{array} \tag{3.40}$$

K algoritmickému výpočtu bázových funkcí $N_{i,p}(u)$ je potřeba nejdříve určit index i, i-tého uzlového vektoru $\langle u_i, u_{i+1}), i = 0, 1, \ldots m$, ve kterém se nachází aktuální hodnota parametru u. Dá se totiž ukázat (viz předchozí příklady či [45]), že na i-tém intervalu $\langle u_i, u_{i+1})$ jsou všechny bázové funkce s výjimkou $N_{i-p,p}, \ldots, N_{i,p}$ rovny nule. Proto je vhodné určit index daného intervalu, abychom nemuseli zbytečně počítat bázové funkce, které jsou zde nulové. Problém při výpočtu může nastat, jestliže $u = u_m$, proto index aktuálního intervalu označíme $n\ (= m - p - 1)$, tedy aktuální hodnota u leží v intervalu $(u_n, u_{n+1}\rangle = (u_{m-p-1}, u_{m-p}\rangle$. K tomuto účelu použijeme následující Algoritmus 3 binárního dělení.

Jakmile jsou spočteny indexy uzlových intervalů pro všechna $u \in \langle 0, 1\rangle$, můžeme vypočítat hodnoty všech nenulových bázových funkcí podle následujícího Algoritmu 4.

Tímto algoritmem tedy určíme všechny nenulové bázové funkce na daném i-tém uzlovém intervalu. Postup aplikujeme na všechny intervaly a tím popíšeme bázové funkce na celém intervalu $\langle 0, 1\rangle$. Tedy pro určení bodů křivky využijeme předchozí algoritmy, které vytvoří Algoritmus 5.

Jako poslední algoritmus výpočtu uvedeme již zmíněný De Boorův algoritmus, který pracuje na základě následující myšlenky. Jak již bylo zmíněno na začátku této kapitoly, s narůstající násobností vnitřních uzlů dochází ke snižování spojitosti křivky v tomto

Algoritmus 3 FindSpan

```
1: procedure FindSpan(n, p, u, U)  ▷ Algoritmus vrátí index intervalu, ve kterém se nachází aktuální parametr u
2:     if (u = U[n + 1]) then return n
3:     end if
4:     low = p
5:     high = n + 1
6:     mid = floor((low + high) / 2)        ▷ zaokrouhlení na dolní celočíselnou hodnotu
7:     while (u < U[mid]) | | (u >= U[mid + 1)] do
8:         if (u < U[mid]) then
9:             high = mid
10:        else
11:            low = mid
12:        end if
13:        mid = floor((low + high) / 2)
14:    end while
15:    return mid
16: end procedure
```

Algoritmus 4 BasisFuns

```
1: procedure BasisFuns(i, u, p, U, N)  ▷ Algoritmus vrátí hodnoty nenulových bázových funkcí N stupně 0 až p v bodě u na příslušném i-tém uzlovém intervalu
2:     N[0] = 1.0
3:     for (j = 1; j <= p; j + +) do
4:         left[j] = u – U[i + 1 – j]
5:         right[j] = U[i + j] – u
6:         saved = 0.0
7:         for (r = 0; r < j; r + +) do
8:             temp = N[r] / (right[r + 1] + left[j – r])
9:             N[r] = saved + right[r + 1] * temp
10:            saved = left[j – r] * temp
11:        end for
12:        N[j] = saved
13:    end for
14:    return N[j]
15: end procedure
```

místě. Současně také dochází ke snižování počtu nenulových bázových funkcí. V uzlovém intervalu $\langle u_i, u_{i+1})$ je počet nenulových bázových funkcí roven $p + 1$. Přímo v uzlu u_i s násobností 1 je tedy p nenulových bázových funkcí, protože v tomto uzlu je koeficient před jednou z nich $u - u_i = u_i - u_i = 0$ (viz rovnice (3.25)). Tudíž s násobností k daného uzlu u_i je počet nenulových bázových funkcí v tomto bodě roven $p - k + 1$. Odtud plyne, že pokud uzlu u_i zvýšíme jeho násobnost na $k = p$, pak počet nenulových funkcí bude $p - k + 1 = p - p + 1 = 1$. Tudíž získáme přímo vyjádření bodu na křivce C v místě u_i, tedy zbylá nenulová funkce $N_{i,p}(u_i) = C(u_i)$.

Algoritmus 5 CurvePoint

```
1: procedure CurvePoint(n, p, U, P, u, C)   ▷ Algoritmus spočítá body na křivce C
2:     span = FindSpan(n, p, u, U)
3:     BasisFuns(span, u, p, U, N)
4:     C = 0.0
5:     for (i = 1; i <= p; i + +) do C =C + N[i] * P[span – p + i]
6:     end for
7:     return C
8: end procedure
```

Tuto myšlenku lze také formulovat takto. Daný uzel u_i vložíme p-krát, tedy docílíme násobnosti p, která způsobí, že dostaneme bod přímo na křivce v hodnotě $u = u_i$. Vše lze také provést p-krát vložením uzlu u_i (Algoritmus 6 – vložení uzlu) do uzlového vektoru, což způsobí přidání nového řídicího bodu. Poslední přidaný řídicí bod bude tedy bodem dané křivky C.

Postup vložení uzlu je popsán v Algoritmu 6, kde k je index vkládaného uzlu (nový uzel je vložen do $\langle u_k, u_{k+1}\rangle$); r počet vložení; s násobnost $(r + s \leq p)$; p stupeň křivky; np počet původních a nq počet nových řídicích bodů; P původní řídicí body a Q nové řídicí body; UP původní uzlový vektor a UQ uzlový vektor po vložení; u parametr křivky. Při volbě $s = 0, r = p$ získáme De Boorův algoritmus.

Algoritmus 6 CurveKnotIns

```
1: procedure CurveKnotIns(np, p ,UP, P, u, k, s, r, nq, UQ, Q) ▷ Algoritmus vrátí
   upravenou křivku (nq, UQ, Q) po vložení nového uzlu
2:     mp = np + p +1
3:     nq = np + r
4:     for (i = 0; i <= k; i + +) do UQ[i] = UP[i]   ▷ načtení nového uzlového vektoru
5:     end for
6:     for (i = 1; i <= r; i + +) do UQ[k + i] = u
7:     end for
8:     for (i = k + 1; i <= mp; i + +) do UQ[i + r] = UP[i]
9:     end for
10:    for (i = 0; i <= k – p; i + +) do Q[i] = P[i]        ▷ uložení nezměněných ř. bodů
11:    end for
12:    for (i = k – s; i <= np; i + +) do Q[i + r] = P[i]
13:    end for
14:    for (i = 0; i <= p – s; i + +) do R[i] = P[k – p + i]
15:    end for
16:    for (j = 1; j <= r; j + +) do                        ▷ r-krát vložení uzlu
17:        L = k – p + j
18:        for (i = 0; i <= p – j – s; i + +) do
19:            alpha = (u – UP[L + i]) / (UP[i + k + 1] – UP[L + i])
20:            R[i] = alpha * R[i + 1] + (1.0 – alpha) * R[i]
21:        end for
22:        Q[L] = R[0]
23:        Q[k + r – j – s] = R[p – j – s]
24:    end for
25:    for (i = L + 1; i < k – s; i + +) do          ▷ načtení zbývajících řídicích bodů
26:        Q[i] = R[i – L]
27:    end for
28:    return nq, UQ, Q
29: end procedure
```

3.3.6. Racionální b-spline křivky

Podobně jako v případě polynomiálních Bézierových křivek, i po částech polynomiální b-spline křivky neumožňují popis například kuželoseček. Tedy i zde se nabízí možnost racionální podoby b-spline křivek (podobně jako v kap. 3.3.4 str. 32).

Racionální b-spline křivky známé pod pojmem *NURBS*, tedy *Non Uniform Rational B-spline*, jsou v současnosti jednou z často používaných matematických forem křivek pro použití ve výpočetní technice. Jsou využívány zejména v grafických programech pro designéry z různých odvětví (od grafického pro průmyslový design). V této kapitole je uveden základní popis NURBS křivek včetně příkladů (více informací v [62], [59], [48]).

Definice 3.30. *NURBS křivka* $C \in \mathbb{R}^2$ stupně p je dána předpisem

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i P_i}{\sum_{i=1}^{n} N_{i,p}(u) w_i}, \ u \in \langle a, b \rangle, \tag{3.41}$$

kde P_i jsou řídicí body (tvořící řídicí polygon), $N_{i,p}(u)$ b-spline bázové (někdy jen bázové) funkce stupně p definované na uzlovém vektoru U ve tvaru

$$U = \{\underbrace{a, \ldots, a}_{p+1}, \underbrace{u_{p+1}, \ldots, u_{m-p-1}}_{n-p}, \underbrace{b, \ldots, b}_{p+1}\}, \tag{3.42}$$

$w_i \in \mathbb{R}$ jsou váhy a reálný interval $\langle a, b \rangle$, který většinou ztotožníme s intervalem $\langle 0, 1 \rangle$.

Váhy $w_i \in \mathbb{B}$ budeme v další části uvažovat pouze kladné, tedy $w_i > 0$. Rovnici lze přepsat do tvaru

$$C(u) = \sum_{i=0}^{n} R_{i,p}(u) P_i, \ u \in \langle 0, 1 \rangle, \text{ kde } R_{i,p}(u) = \frac{N_{i,p}(u) w_i}{\sum_{j=0}^{n} N_{j,p}(u) w_j}, \tag{3.43}$$

přičemž $R_{i,p}(u)$ jsou *racionální b-spline bázové funkce*, které jsou po částech polynomiální na intervalu $\langle 0, 1 \rangle$. Podobně jako v případě racionálních Bézierových křivek, i zde mají všechny racionální b-spline bázové funkce stejný jmenovatel. Navíc při volbě $w_i = 1, i = 0, 1, \ldots, n$ obdržíme vyjádření klasické b-spline křivky (viz kap. 3.3.5 str. 34).

Příklad 3.31. Pro jednoduchost mějme kvadratickou racionální b-spline křivku určenou řídicím polygonem P_0, P_1 a P_2, uzlovým vektorem $U = \{0, 0, 0, 1, 1, 1\}$ a vahami $w_0 = 1$, $w_1 = 4$ a $w_2 = 1$ (obr. 3.36). Jednotlivé b-spline bázové funkce stupně 2 jsou tvaru (viz př. 3.27)

$$N_{0,2}(u) = \frac{u - u_0}{u_2 - u_0} N_{0,1} + \frac{u_3 - u}{u_3 - u_1} N_{1,1} = \frac{u - 0}{0 - 0} N_{0,1} + \frac{1 - u}{1 - 0} N_{1,1} = \begin{cases} (1-u)^2, & u \in \langle 0, 1) \\ 0 & \text{jinde} \end{cases} \tag{3.44}$$

$$N_{1,2}(u) = \frac{u - u_1}{u_3 - u_1} N_{1,1} + \frac{u_4 - u}{u_4 - u_2} N_{2,1} = \frac{u - 0}{1 - 0} N_{1,1} + \frac{1 - u}{1 - 0} N_{2,1} = \begin{cases} 2u(1-u), & u \in \langle 0, 1) \\ 0 & \text{jinde} \end{cases} \tag{3.45}$$

$$N_{2,2}(u) = \frac{u-u_2}{u_4-u_2}N_{2,1} + \frac{u_5-u}{u_5-u_3}N_{3,1} = \frac{u-0}{1-0}N_{1,1} + \frac{1-u}{1-1}N_{2,1} = \begin{cases} u^2, & u \in \langle 0,1) \\ 0 & \text{jinde.} \end{cases} \tag{3.46}$$

společný jmenovatel racionálních b-spline bázových funkcí je tedy tvaru

$$\begin{aligned} \textstyle\sum_{j=0}^{n} N_{j,p}(u)w_j &= N_{0,2}(u)w_0 + N_{1,2}(u)w_1 + N_{2,2}(u)w_2 = \\ &= (1-u)^2 \cdot 1 + 2u(1-u)\cdot 4 + u^2 \cdot 1 = \\ &= -6u^2 + 6u + 1, \ u \in \langle 0,1\rangle \end{aligned} \tag{3.47}$$

a racionální b-spline funkce mají tvar

$$R_{0,2}(u) = \frac{(1-u)^2\cdot 1}{-6u^2+6u+1},\ R_{1,2}(u) = \frac{2u(1-u)\cdot 4}{-6u^2+6u+1},\ R_{2,2}(u) = \frac{u^2\cdot 1}{-6u^2+6u+1},\ u \in \langle 0,1\rangle, \tag{3.48}$$

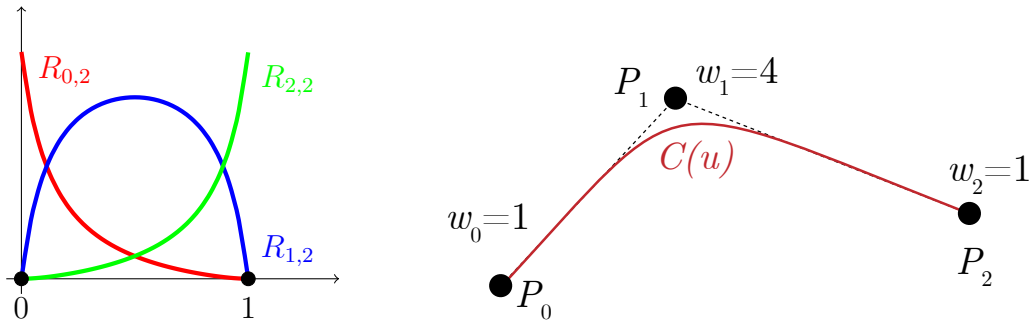
pak takto určená racionální b-spline křivka má rovnici

$$C(u) = \frac{(1-u)^2}{-6u^2+6u+1}P_0 + \frac{8u(1-u)}{-6u^2+6u+1}P_1 + \frac{u^2}{-6u^2+6u+1}P_2,\ u \in \langle 0,1\rangle. \tag{3.49}$$

Věta 3.32. Racionální bázové funkce $R_{i,n}(t)$ mají tyto klíčové vlastnosti

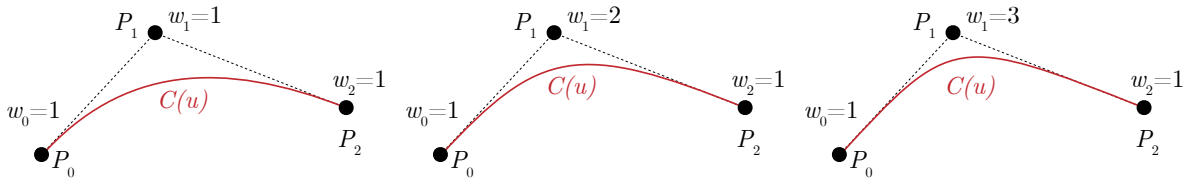
- $\forall i,n:\ R_{i,n}(u) \geq 0,\ u \in \langle 0,1\rangle$,
- $\sum_{i=0}^{n} R_{i,n}(u) = 1,\ u \in \langle 0,1\rangle$,
- $R_{0,0}(0) = R_{n,n}(1) = 1$.

Důkaz lze najít v [45].



(a) (b)

Obrázek 3.36: Kvadratická racionální b-spline křivka (a) racionální bázové b-spline funkce, (b) křivka.



(a) (b) (c)

Obrázek 3.37: Kvadratická racionální b-spline křivka s různými vahami w_1 (a) $w_1 = 1$, (b) $w_1 = 2$, (c) $w_1 = 3$.

Výpočet b-spline bázových funkcí

Pro výpočet b-spline bázových funkcí použijeme algoritmy z kap. 3.3.5 (str. 34), jmenovitě algoritmy **FindSpan** a **BasisFuns** a jejich kombinací je následující Algorimtus 7. Nechť Pw obsahuje řídicí body racionální b-spline křivky, tedy Pw[i] $= P_i = (w_i x_i, w_i y_i, w i_z i, w_i)$, Cw označme bod křivky $C^w(u)$ včetně jeho váhy (tedy 4 rozměrný bod) a C bod křivky $C(u)$. Při práci s váhami přecházíme do projektivního prostoru, tedy souřadnice bodů mají 4 souřadnice. Více o zavedení a významu homogenních souřadnic lze nalézt v [38].

Algoritmus 7 RCurvePoint

```
1:  procedure RCurvePoint(n, p, U, Pw, u, C)      ▷ Algoritmus vypočítá body na
    racionální křivce C
2:      span = FindSpan(n, p, u, U)
3:      BasicFuns(span, u, p, U, N)
4:      Cw = 0.0
5:      for (j = 0; j<= p; j + +) do
6:          Cw = Cw + N[j] * Pw[span – p + j]
7:          C = Cw / w
8:      end for
9:      return C
10: end procedure
```

3.4. Prokládaní dat vhodnými křivkami

Curve fitting, neboli *proložení křivky* zadanými body je problematika *matematické regrese*. Většinou se setkáváme s empiricky naměřenými hodnotami $y_i \in \mathbb{R}, i = 0, 1, 2, \ldots n$ během experimentu x_i (teplota maziva při stoupajících otáčkách motoru, nárůst počtu obyvatel v průběhu semestru, cena akcií na burze, či v případě mračen bodů – poloha bodu v prostoru), jejichž chování (polohu) chceme popsat matematickým modelem – funkcí – křivkou.

Princip nalezení takového modelu lze obecně rozdělit do dvou kategorií: *interpolace* a *aproximace* [11], [57]. V případě interpolace hledáme funkci f, která přímo prochází naměřenými hodnotami y_i, tedy $f(x_i) = y_i$, kde $i = 0, 1, 2 \ldots n$. Mezi nejpoužívanější metody interpolace patří: *interpolace polynomem* a *interpolace spline*.

V druhém případě aproximace, hledáme funkci f která se přibližuje (aproximuje) s určitou chybou $E \in \mathbb{R}^+$ k naměřeným hodnotám. Tedy $\forall x_i, i = 0, 1, 2 \ldots n; |y_i - f(x_i)| \leq E$.

V této části textu jsou uvedeny základní matematické principy interpolace a aproximace a posléze jejich aplikace pro použití prokladu křivek v mračnu bodů.

3.4.1. Interpolace polynomem

Definice 3.33. Mějme dány navzájem různé body $x_i \in \mathbb{R}, i = 0, 1, 2, \ldots n, x_i \neq x_j$ pro $i \neq j$, tzv. *uzly*, ve kterých jsou naměřeny hodnoty y_i. *Interpolační polynom stupně nejvýše* n, $P_n(x)$ je polynom, který splňuje tzv. *interpolační podmínky*

$$P_n(x_i) = y_i, \text{ pro } i = 0, 1, 2 \ldots n. \tag{3.50}$$

Lagrangeův interpolační polynom

Definice 3.34. *Lagrangeův interpolační polynom stupně* n, $L_n(x)$ je ve tvaru

$$L_n(x) = y_0 l_0(x) + y_1 l_1(x) + \cdots y_n l_n(x) = \sum_{i=0}^{n} y_i l_i(x), \tag{3.51}$$

kde $l_i(x)$ jsou tzv. *fundamentální polynomy* ve tvaru

$$l_i(x) = \frac{(x - x_0)(x - x_1) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0)(x_i - x_1) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)}, \tag{3.52}$$

kde zřejmě

$$l_i(x_k) = \begin{cases} 1 & \text{pro } k = i \\ 0 & \text{pro } k \neq i, \end{cases} \quad \text{kde } i, k = 0, 1, \ldots n, \tag{3.53}$$

tedy interpolační podmínky $P_n(x_k) = \sum_{i=0}^{n} y_i l_i(x_k) = y_k$ jsou splněny.

Lagrangeův interpolační polynom má výhodu v přehledné a výpočetně nenáročné konstrukci. Avšak přidání uzlu znamená přepočtení všech fundamentálních polynomů, což je značná nevýhoda pro praktické využití. Nevýhodu tohoto přístupu eliminuje následující metoda.

Newtonův interpolační polynom

Definice 3.35. *Newtonův interpolační polynom stupně n, $N_n(x)$* je ve tvaru

$$N_n(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + \cdots + a_n(x-x_0)(x-x_1)\cdots(x-x_{n-1}), \quad (3.54)$$

kde koeficienty a_i lze určit buď z interpolačních podmínek, nebo pomocí tzv. *poměrných diferencí* P ve tvaru

$$\begin{aligned} &P[x_i] && = y_i \\ &P[x_i, x_{i+1}] && = (P[x_{i+1}] - P[x_i])/(x_{i+1} - x_i), \\ &P[x_i, x_{i+1}, x_{i+2}] && = (P[x_{i+1}, x_{i+2}] - P[x_i, x_{i+1}])/(x_{i+2} - x_i), \\ &\text{pro } 3 \le k \le n: && \\ &P[u_i, x_{i+1}, \ldots, x_{i+k}] && = (P[x_{i+1}, \ldots, x_{i+k}] - P[x_i, \ldots, x_{i+k-1}])/(x_{i+k} - x_i), \end{aligned} \quad (3.55)$$

kde lze ověřit, že $a_i = P[x_0, x_1, \ldots, x_i]$. Položíme-li $P_{ik} = P[x_{i-k}, \ldots, x_i]$, pak $a_i = P_{ii}$ a $P_{i0} = y_i$, pak Newtonův interpolační polynom přejde do tvaru

$$N_n(x) = P_{00} + P_{11}(x-x_0) + P_{22}(x-x_0)(x-x_1) + \cdots + P_{nn}(x-x_0)(x-x_1)\cdots(x-x_{n+1}). \quad (3.56)$$

Výpočet poměrných diferencí P_{ii} lze přehledně zapsat do schematu, které se vyplňuje po sloupcích.

$$\begin{array}{c|cccccc} u_0 & P_{00} & & & & \\ u_1 & P_{10} & P_{11} & & & \\ u_2 & P_{20} & P_{21} & P_{22} & & \\ \vdots & \vdots & \vdots & & \ddots & \\ u_n & P_{n0} & P_{n1} & \cdots & P_{n,n-1} & P_{nn}. \end{array} \quad (3.57)$$

Jestliže přidáme další uzel u_{n+1}, stačí dopočíst poměrnou diferenci $P_{n+1,n+1}$ a k původnímu polynomu přičíst další člen, tedy

$$N_{n+1}(x) = N_n(x) + P_{n+1,n+1}(x-x_0)(x-x_1)\cdots(x-x_n)(x-x_{n+1}). \quad (3.58)$$

Hermitův interpolační polynom

V případě, že v jednotlivých uzlech $u_i,\ i = 0, 1, 2, \ldots, n$ jsou zadány také derivace až do určitého řádu $\alpha_i + 1$, tedy $y^{(0)}, y^{(1)}, \ldots, y^{(\alpha_i)}$, pak mluvíme o tzv. *Hermitově interpolačním polynomu*. Nyní je uvedena definice využitím interpolačních podmínek. Podrobnější definici lze najít například v [57].

Definice 3.36. Nechť $\alpha = n + \sum_{i=0}^{n} \alpha_i$, pak *Hermitův interpolační polynom stupně nejvýše* α je polynom $H_\alpha(x)$, který splňuje následující interpolační podmínky

$$\frac{\mathrm{d}^j}{\mathrm{dx}^j} H_\alpha(x_i) = y_i^j, \text{ pro } j = 0, 1, 2, \ldots, \alpha_i \text{ a } i = 0, 1, 2, \ldots, n. \tag{3.59}$$

Podobně jako v případě polynomiálních křivek (viz kap. 3.3.2 str. 28), se zde potýkáme se stoupající výpočetní náročností způsobenou vysokým stupněm polynomu pro proklad velkého počtu vstupních dat. Proto i zde je vhodné body prokládat po jednotlivých polynomiálních částech nižšího stupně, tedy pomocí spline funkcí.

3.4.2. Interpolace spline

Mějme dány různé body – uzly $x_i \in \mathbb{R}, i = 0, 1, 2, \ldots, n$ rozdělující interval $\langle a, b\rangle \in \mathbb{R}$

$$a = u_0 < x_1 < \cdots < x_{i-1} < x_i < x_{i+1} < \cdots < x_{n-1} < x_n = b, \tag{3.60}$$

s hodnotami $y_i \in \mathbb{R}$, které chceme interpolovat polynomem na jimi určeném intervalu $\langle x_0, x_n\rangle$. Klasickým interpolačním postupem bychom získali polynom s vysokým stupněm (nejvýše n), který je výpočetně náročný a vykazuje velké chyby mezi uzly. Proto je vhodnější vstupní data prokládat postupně po částech polynomiálními – tzv. *spline funkcemi* nízkého stupně. Jejich konstrukce je intuitivní a výpočetně méně náročná, a proto se ve velké míře vyskytují v algoritmech užívaných ve výpočetní technice.

Hledaný interpolační spline $S(x), x \in \langle a, b\rangle$ je tedy složen z $n-1$ dílčích polynomů $S_i(x), x \in \langle x_{i-1}, x_i\rangle, i = 1, 2, \ldots, n$ na intervalu mezi uzly x_{i-1} a x_i. Přičemž interpolační podmínky $S_i(x_{i-1}) = y_{i-1}$ a $S_i(x_i) = y_i$ jsou na dílčích polynomech zřejmě splněny. Dále také platí $S_i(x_i) = S_{i+1}(x_i), i = 1, 2, \ldots, n-1$.

Délku i-tého intervalu $\langle x_{i-1}, x_i\rangle$ označíme h_i a pro zjednodušení zápisu definic jednotlivých interpolačních spline zavedeme tato označení

$$s = x - x_{i-1} \tag{3.61}$$

pro lokální rozdíl mezi uzly,

$$\delta_i = \frac{y_i - y_{i-1}}{x_i - x_{i-1}} = \frac{y_i - y_{i-1}}{h_i} \tag{3.62}$$

pro poměrnou diferenci (směrnici přímky) mezi sousedními body.

V následující části textu je uveden lineární interpolační a kubický spline (více v [11], [57]).

Lineární interpolační spline

Intuitivně nejjednodušší způsob sestrojení interpolačního spline je spojení dvou sousedních bodů $[x_{i-1}, y_{i-1}]$ a $[x_i, y_i]$ přímkou. Získáme tedy tzv. *lineární interpolační spline.*

Definice 3.37. *Lineární interpolační spline* je dán dílčími polynomy $S_i(x), i = 1, 2, \ldots, n$

$$S_i(x) = y_{i-1} + \frac{y_i - y_{i-1}}{x_i - x_{i-1}}(x - x_{i-1}) = y_{i-1} + s\delta_i, \text{ kde } x \in \langle x_{i-1}, x_i\rangle. \tag{3.63}$$

Hermitův kubický spline

Definice 3.38. *Hermitův kubický spline* je spline funkce $S(x)$ na intervalu $\langle a, b\rangle$ daná dílčími polynomy $S_i(x), i = 1, 2, \ldots, n$

$$S_i(x) = y_{i-1} + sd_{i-1} + s^2\frac{3\delta_i - 2d_{i-1} - d_i}{h_i} + s^3\frac{d_{i-1} - 2\delta_i + d_i}{h_i^2}, \tag{3.64}$$

která splňuje následující podmínky

1. $S(x) \in C^1\langle a, b\rangle$,
2. $S(x_i) = y_i, i = 0, 1, 2, \ldots, n$,
3. $S'(x_i) = d_i, i = 0, 1, 2 \ldots, n$,
4. na každém intervalu $\langle x_{i-1}, x_i\rangle, i = 1, 2, \ldots, n$ je polynomem nejvýše třetího stupně

kde y_i jsou zadané funkční hodnoty a d_i jsou zadané derivace v jednotlivých uzlech.

3.4.3. Interpolace b-spline křivek

Algoritmy interpolace b-spline křivek lze rozdělit do dvou kategorií: *globální* a *lokální*. V případě globálních algoritmů se sestaví soustava rovnic pro celková data, která se následně vyřeší. Jestliže jsou neznámé pouze řídicí body, systém rovnic je lineární a snadno řešitelný. V jiném případě, kdy je například určena křivost, je systém již nelineární. V globálním případě může změna jednoho parametru ovlivnit celkový vzhled křivky. Oproti tomu, lokální algoritmy řeší soustavy rovnic vždy pro danou část dat v určeném pořadí. Změna parametrů se projeví pouze lokálně v určeném segmentu. Lokální metody jsou také výpočetně méně náročné, avšak pro zajištění požadované spojitosti mezi jednotlivými segmenty jsou vhodnější metody globální.

V této části je popsán základní algoritmus globální interpolace b-spline křivky (více v [45],[27])

Globální b-spline interpolace

Mějme dány body $Q_k \in \mathbb{R}^3, k = 0, 1, 2, \ldots, n$, kterými chceme proložit b-spline křivku stupně p ve tvaru

$$Q_k = C(\bar{u}_k) = \sum_{i=1}^{n} N_{i,p}(\bar{u}_k)P_i, \text{ kde } \bar{u}_k \in \langle 0, 1\rangle, \tag{3.65}$$

kde $\bar{u}_k$ jsou přiřazené parametry pro příslušné body, tedy $C(\bar{u}_k) = Q_k, k = 0, 1, 2, \ldots, n$, $N_{i,p}(\bar{u}_k)$ jsou b-spline bázové funkce a P_i řídicí body (viz kap. 3.3.5 str. 34). Pro určení této křivky je třeba vhodně zvolit hodnoty $\bar{u}_k$, uzlový vektor $U = \{u_0, u_1, \ldots, u_m\}$ a následně vyřešit rovnici (3.65) pro $n + 1$ neznámých řídicích bodů P_i.
Existují tři základní metody pro volbu parametrů $\bar{u}_k \in \langle 0, 1\rangle$.

1. **metoda ekvidistantního rozdělení**

$$\begin{aligned} \bar{u}_0 &= 0 \quad \bar{u}_n = 1 \\ \bar{u}_k &= \tfrac{k}{n} \quad \text{pro } k = 1, 2, \ldots, n-1. \end{aligned} \tag{3.66}$$

Tato metoda je vhodná pouze pro rovnoměrně rozdělené vstupní body. V jiném případě může vytvářet smyčky nebo nevhodné tvary.

2. **metoda tětivy**
   Nechť d je celková délka tětivy proložené mezi všemi body Q_k

$$d = \sum_{k=1}^{n} |Q_k - Q_{k-1}|, \tag{3.67}$$

   pak

$$\begin{aligned} &\bar{u}_0 = 0 && \bar{u}_n = 1 \\ &\bar{u}_k = \bar{u}_{k-1} + \tfrac{|Q_k - Q_{k-1}|}{d} && \text{pro } k = 1, 2, \ldots, n-1. \end{aligned} \tag{3.68}$$

   Jde o nejčastěji používanou metodou a produkuje přijatelné výsledky.

3. **metoda dostředivé síly**
   Nechť

$$d = \sum_{k=1}^{n} \sqrt{|Q_k - Q_{k-1}|}, \tag{3.69}$$

   pak

$$\begin{aligned} &\bar{u}_0 = 0 && \bar{u}_n = 1 \\ &\bar{u}_k = \bar{u}_{k-1} + \tfrac{\sqrt{|Q_k - Q_{k-1}|}}{d} && \text{pro } k = 1, 2, \ldots, n-1. \end{aligned} \tag{3.70}$$

   Pro data s ostrými záhyby tato metoda (více v [36]) podává lepší výsledky než metoda tětivy.

Pro volbu uzlového vektoru lze použít rovnoměrné rozdělení nebo průměrování.

1. **rovnoměrné rozdělení**

$$\begin{aligned} &u_0 = \cdots = u_p = 0 && u_{m-p} = \cdots = u_m = 1 \\ &u_{j+p} = \tfrac{j}{n-p+1} && \text{kde } j = 1, 2, \ldots, n-p \end{aligned} \tag{3.71}$$

   Tato metoda se však nedoporučuje, protože rovnice (3.65) může vést k singulárnímu systému rovnic.

2. **průměrování**

$$\begin{aligned} &u_0 = \cdots = u_p = 0 && u_{m-p} = \cdots = u_m = 1 \\ &u_{j+p} = \tfrac{1}{p} \textstyle\sum_{j=1}^{j+p-1} \bar{u}_i && \text{kde } j = 1, 2, \ldots, n-p \end{aligned} \tag{3.72}$$

   Tato metoda s kombinací s metodou tětivy či dostředivé síly vede k přijatelnému řešení systému (3.65), kde $N_{i,p}(\bar{u}_k) = 0$ pro $|i - k| \geq p$. Sytém rovnic pak řešit Gaussovou eliminační metodou bez pivotování a tím vypočítat neznáme řídicí body P_i, které b-spline křivku jednoznačně určí.

3.4.4. Aproximace metodou nejmenších čtverců

Metoda nejmenších čtverců (v angličtině *least square*) je regresní statistickou metodou. Jde o metodu, která pro zadanou množinu naměřených hodnot určí parametry zvoleného matematického modelu, který aproximačně popisuje jejich průběh [11]. Jinak řečeno, metoda nejmenších čtverců se používá pro prokládání dat přímkami, křivkami či polynomy vyšších stupňů, kde se minimalizuje součet kvadrátů residuí naměřených hodnot od zvoleného modelu. Odtud také název metody.

Kromě této aplikace se nejmenší čtverce aplikují i pro řešení tzv. *přeurčených* soustav lineárních rovnic, kde je počet rovnic větší než počet proměnných.

Metoda nejmenších čtverců

Princip metody dle [11] je následující. Mějme m naměřených hodnot y_i v různých časech t_i, kde $i = 1, 2, \dots m$. Budeme předpokládat, že neznámá funkce $y(t)$ lze popsat pomocí vhodně zvolené funkce $R_n(t)$ dané *n–bázovými funkcemi* φ_i pro některé $n \leq m$. Funkce $R_i(t)$ se ve statistice označuje jako *lineární regresní funkce* a lze ji zapsat takto:

$$y(t) \approx x_1\varphi_1(t) + x_2\varphi_2(t) + \cdots + x_n\varphi_n(t) := R_n(t). \tag{3.73}$$

Úkolem je určit parametry x_i navrhované funkce $R_i(t)$ s volbou bázových funkcí φ_i podle předpokládaného průběhu.

Z naměřených hodnot a bázových funkcí se vytvoří tzv. *návrhová matice* A typu $m \times n$:

$$A = \begin{pmatrix} \varphi_1(t_1) & \varphi_2(t_1) & \dots & \varphi_n(t_1) \\ \varphi_1(t_2) & \varphi_2(t_2) & \dots & \varphi_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(t_m) & \varphi_2(t_m) & \dots & \varphi_n(t_m) \end{pmatrix} = (\phi_1, \phi_2, \dots, \phi_n), \tag{3.74}$$

kde $\phi_i = (\varphi_i(t_1), \varphi_i(t_2), \dots, \varphi_i(t_m))^T$a maticová rovnice modelu je tvaru:

$$y \approx Ax, \tag{3.75}$$

kde $y = (y_1, y_2, \dots, y_m)^T$ jsou naměřená data a $x = (x_1, x_2, \dots, x_n)^T$ je vektor neznámých.

Hledaná a navržená funkce mají v každém bodě x_i rozdílné funkční hodnoty zvané *residua* r_i ve tvaru:

$$r_i = y_i - R_n(t_i) = y_i - \sum_{j=1}^{n} \varphi_j(t_i)x_j = y_i - \sum_{j=1}^{n} a_{ij}x_j \text{ , kde } a_{ij} = \varphi_j(t_i). \tag{3.76}$$

Maticově zapsáno:

$$r = y - Ax. \tag{3.77}$$

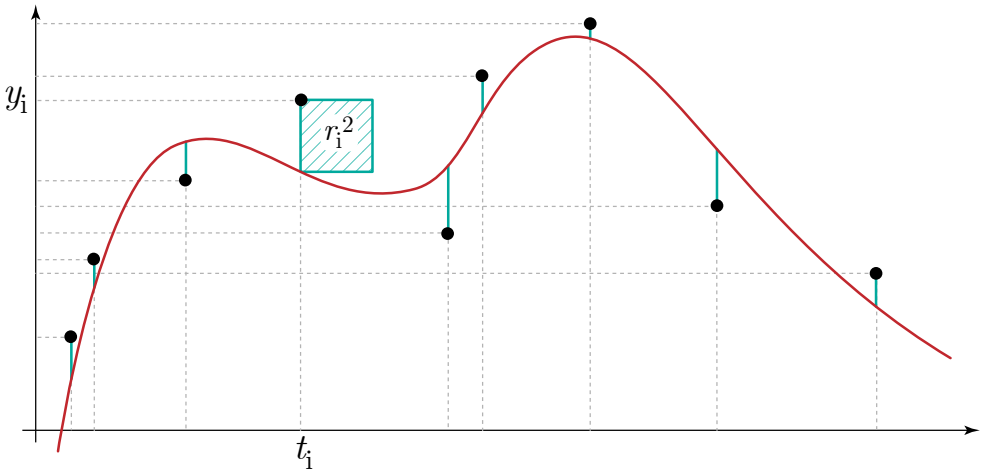
Při řešení situace metodou nejmenších čtverců chceme, aby residua byla co nejmenší a tedy minimalizujeme součet jejich druhých mocnin (součet čtverců, viz obr. 3.38).

$$\|r\|^2 := \sum_{i=1}^{m} r_i^2 \longrightarrow \min \tag{3.78}$$

Jestliže jsou některá měření přesnější, lze použít i *váženou* metodu

$$\|r\|_w^2 := \sum_{i=1}^{m} w_i r_i^2 \longrightarrow \min, \tag{3.79}$$

kde w_i je váha hodnoty y_i, $i = 1, 2, \ldots, m$.



Obrázek 3.38: Grafické znázornění principu metody nejmenších čtverců v bodě t_i.

Minimalizace residuí se provede přes extrém funkce z rovnice (3.78). Rovnice se parciálně derivuje dle jednotlivých proměnných x_i, čímž se spočte gradient této funkce a položí se roven 0. Po úpravách dostaneme soustavu lineárních rovnic v maticovém tvaru:

$$A^T A x = A^T y \equiv Gx = A^T y, \tag{3.80}$$

která se nazývá *normální soustava rovnic*. Matice $G := A^T A$ se nazývá *Grammova matice*. Za podmínky, že řádky matice A jsou lineárně nezávislé, má tato soustava jediné řešení x^* a platí:

$$\|y - Ax^*\|^2 = \min_x \|y - Ax\|^2. \tag{3.81}$$

Rovnice (3.80) má při dosazení vektorů ϕ_i tvar:

$$\begin{pmatrix} (\phi_1,\phi_1) & (\phi_1,\phi_2) & \cdots & (\phi_1,\phi_n) \\ (\phi_2,\phi_1) & (\phi_2,\phi_2) & \cdots & (\phi_2,\phi_n) \\ \vdots & \vdots & \ddots & \vdots \\ (\phi_n,\phi_1) & (\phi_n,\phi_2) & \cdots & (\phi_n,\phi_n) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} (\phi_1, y) \\ (\phi_2, y) \\ \vdots \\ (\phi_n, y) \end{pmatrix}, \tag{3.82}$$

kde

$$(\phi_j, \phi_k) = \sum_{i=1}^{m} \phi_j(t_i)\phi_k(t_i) \text{ a } (\phi_j, y) = \sum_{i=1}^{m} \phi_j(t_i) y_i \tag{3.83}$$

jsou skalární součiny. Výpočtem neznámých $x_1, x_2, \ldots, x_n$ z rovnice (3.82) získáme hledané parametry funkce $R_n(t)$.

3.4.5. Aproximace b-spline křivek

Aproximovat vstupní data b-spline křivkou je náročnější než její interpolace, kdy se jednoduše určí počet řídicích bodů a uzlový vektor. V případě aproximace známe jen vstupní body a požadovanou maximální chybu $E \in \mathbb{R}$. Metody aproximace lze opět rozdělit do dvou kategorií: globální a lokální (viz kap. 3.4.3 str. 48).
Globální algoritmy pracují obvykle iterativně dvěma způsoby.

1. **Vstup je dán minimálním počtem řídicích bodů a algoritmus následně:**
   (a) proloží křivku daty
   (b) porovná odchylku křivky od všech bodů s maximální hodnotou E
   (c) pokud je odchylka splněna, vrátí nalezenou křivku; jinak se zvýší počet řídicích bodů a přejde se k prvnímu kroku
2. **Vstup je dán maximálním počtem řídicích bodů a algoritmus následně:**
   (a) proloží křivku daty
   (b) porovná odchylku křivky od všech bodů s maximální hodnotou E
   (c) pokud je odchylka splněna, vrátí nalezenou křivku; jinak se sníží počet řídicích bodů a přejde se k prvnímu kroku

V této části je uveden základní algoritmus globální aproximace b-spline křivky využitím metody nejmenších čtverců (více o těchto metodách v [45]). Aproximaci b-spline křivky používáme v navrženém algoritmu (viz kap. 4.4 str. 81 věnovaná výsledkům disertační práce).

Globální b-spline aproximace metodou nejmenších čtverců

Mějme dán stupeň $p \geq 1$ hledané křivky, dále $n \geq p$ a vstupní body $Q_k \in \mathbb{R}^3, k = 0, 1, 2, \ldots, m$, kde $m > n$. Hledáme b-spline křivku stupně p ve tvaru

$$C(u) = \sum_{i=0}^{n} N_{i,p}(u) P_i, \text{ kde } u \in \langle 0, 1 \rangle, \tag{3.84}$$

kde $N_{i,p}(u)$ jsou b-spline bázové funkce, P_i jsou neznámé řídicí body, a která splňuje $Q_0 = C(0)$ a $Q_m = C(1)$. Pro vypočtení neznámých řídicích bodů P_i je potřeba určit parametry $\{\bar{u}_k\} \in \langle 0.1 \rangle$ a uzlový vektor $U = \{u_0, u_1, \ldots, u_m\}$.

Určení parametrů $\bar{u}_k$

Množinu parametrů $\bar{u}_k$ určíme pomocí metody tětiv (viz kap. 3.4.3 str. 48) tedy

$$\begin{aligned} \bar{u}_0 &= 0 & \bar{u}_n &= 1 \\ \bar{u}_k &= \bar{u}_{k-1} + \tfrac{|Q_k - Q_{k-1}|}{d} & &\text{pro } k = 1, 2, \ldots, n-1, \end{aligned} \tag{3.85}$$

kde

$$d = \sum_{k=1}^{n} |Q_k - Q_{k-1}|. \tag{3.86}$$

Určení uzlového vektoru U

Uzlový vektor $U = \{u_0, u_1, \ldots, u_m\}$ odrážející rozdělení parametrů $\{\bar{u}_k\}$, kde celkový počet potřebných uzlů je $n + p + 2$ je ve tvaru

$$U = \{\underbrace{u_0, \ldots, u_p}_{p+1}, \underbrace{u_{p+1}, \ldots, u_{m-p-1}}_{n-p}, \underbrace{u_{m-p}, \ldots, u_m}_{p+1}\}. \tag{3.87}$$

Označme

$$d = \frac{m+1}{n-p+1}, \tag{3.88}$$

pak vnitřní uzly určíme následovně.

$$\begin{aligned} &\text{nechť } i = int(jd) && \alpha = jd - i, \text{ pak} \\ &u_{p+j} = (1-\alpha)\bar{u}_{i-1} + \alpha\bar{u}_i && \text{pro } j = 1, 2, \ldots, n-p. \end{aligned} \tag{3.89}$$

Výpočet řídicích bodů P_i hledané b-spline křivky

V předchozích krocích jsme určili parametry $\{\bar{u}_k\}$, odpovídající uzlový vektor $U = \{u_0, u_1, \ldots, u_m\}$. Ze vstupních bodů známe hodnoty $Q_0 = C(0)$ a $Q_m = C(1)$. Zbylými body $Q_k, k = 1, 2, \ldots, m-1$ křivka $C(\bar{u})$ přímo neprochází, ale pouze se k nim přibližuje. Neznámé řídicí body P_i tudíž určíme pomocí minimalizace vzdálenosti zadaných bodů Q_k od aproximované křivky $C(\bar{u}_k)$, tedy metodou nejmenších čtverců minimalizujeme tzv. *cílovou funkci* f (v angličtině *objective function*)

$$f(P_i) = \sum_{k=1}^{m-1} |Q_k - C(\bar{u}_k)|^2, \tag{3.90}$$

vzhledem k $n+1$ neznámým P_i.

Jak bylo zmíněno výše, hledaná křivka body $Q_k, k = 1, 2, \ldots, m-1$ přímo neprochází. V cílové funkci f (3.90) nahradíme Q_k bodem R_k

$$R_k = Q_k - N_{0,p}(\bar{u}_k)Q_0 - N_{n,p}(\bar{u}_k)Q_m, \text{ pro } k = 1, 2, \ldots, m-1, \tag{3.91}$$

která pak přejde do tvaru

$$\sum_{k=1}^{m-1} |R_k - C(\bar{u}_k)|^2 = \sum_{k=1}^{m-1} |R_k - \sum_{i=1}^{n-1} N_{i,p}(\bar{u}_k)P_i|^2. \tag{3.92}$$

Po algebraických úpravách funkce f získáme

$$f = \sum_{k=1}^{m-1} \left[R_k \cdot R_k - 2\sum_{i=1}^{n-1} N_{i,p}(\bar{u}_k)(R_k \cdot P_i) + \left(\sum_{i=1}^{n-1} N_{i,p}(\bar{u}_k)P_i\right) \cdot \left(\sum_{i=1}^{n-1} N_{i,p}(\bar{u}_k)P_i\right) \right]. \tag{3.93}$$

Standardní metodou nejmenších čtverců ([11], [14], [61], [35]) tuto funkci f minimalizujeme pro $n-1$ proměnných $P_1, P_2, \ldots, P_{n-1}$. Zbývající řídicí body P_0 a P_n jsou ztotožněny s prvním a posledním zadaným bodem, tedy $P_0 = Q_0$ a $P_n = Q_m$.

Určíme tedy derivaci funkce f podle proměnné P_l

$$\frac{\partial f}{\partial P_l} = \sum_{k=1}^{m-1} \left(-2N_{l,p}(\bar{u}_k)R_k + 2N_{l,p}(\bar{u}_k) \sum_{i=1}^{n-1} N_{i,p}(\bar{u}_k)P_i \right). \tag{3.94}$$

Derivaci položíme rovnu 0 a upravíme na tvar

$$-\sum_{k=1}^{m-1} N_{l,p}(\bar{u}_k)R_k + \sum_{k=1}^{m-1}\sum_{i=1}^{n-1} N_{l,p}(\bar{u}_k)N_{i,p}(\bar{u}_k)P_i = 0, \tag{3.95}$$

ze kterého zřejmě platí

$$\sum_{i=1}^{n-1} \left(\sum_{k=1}^{m-1} N_{l,p}(\bar{u}_k)N_{i,p}(\bar{u}_k) \right) P_i = \sum_{k=1}^{m-1} N_{l,p}(\bar{u}_k)R_k. \tag{3.96}$$

Tato rovnice je lineární s neznámými $P_1, P_2, \ldots, P_{n-1}$ a pro $l = 1, 2, \ldots, n-1$ vytváří systém $(n-1)$ rovnic o $(n-1)$ neznámých, což lze maticově zapsat následně

$$(N^T N)P = R, \tag{3.97}$$

kde N je matice skalárů typu $(m-1) \times (n-1)$

$$N = \begin{pmatrix} N_{1,p}(\bar{u}_1) & \cdots & N_{n-1,p}(\bar{u}_1) \\ \vdots & \ddots & \vdots \\ N_{1,p}(\bar{u}_{m-1}) & \cdots & N_{n-1,p}(\bar{u}_{m-1}) \end{pmatrix}, \tag{3.98}$$

R je vektor $n-1$ bodů tvaru

$$R = \begin{pmatrix} N_{1,p}(\bar{u}_1)R_1 + \cdots + N_{1,p}(\bar{u}_{m-1})R_{m-1} \\ \vdots \\ N_{n-1,p}(\bar{u}_1)R_1 + \cdots + N_{n-1,p}(\bar{u}_{m-1})R_{m-1} \end{pmatrix} \tag{3.99}$$

a P je vektor neznámých $P_1, P_1, \ldots, P_{n-1}$

$$P = \begin{pmatrix} P_1 \\ \vdots \\ P_{n-1}. \end{pmatrix}. \tag{3.100}$$

Tuto rovnici (3.97) při předchozí volbě parametrů $\{\bar{u}_k\}$ a uzlovém vektoru $U = \{u_1, \ldots, u_m\}$ lze řešit Gausovou eliminační metodou. Tím získáme řídicí body P_i a počáteční b-spline křivka je tedy jednoznačně určena. Následně se určí chyba aproximace $e_i \in \mathbb{R}$ – vzdálenost Q_i a $C(\bar{u}_i), i = 0, 1, 2, \ldots, n$, která se porovná s určenou chybou $E \in \mathbb{R}^+$. Jestliže je maximální chyba v rámci tolerance, tedy $\max_i (|e_i|) \leq E$, pak algoritmus vrátí aktuální křivku. Pokud je hodnota mimo určenou toleranci, přidá se další uzel a vyřeší se upravená cílová funkce, než je maximální chyba v mezi tolerance.

V praxi se setkáváme s upravenou cílovou funkcí f (3.97). Ta je většinou upravena vyhlazovací funkcí, aby měla nalezená křivka přijatelný tvar. Cílová funkce je tedy tvaru

$$f(P_j) = \frac{1}{2}\sum_{i=0}^{m} {e_i}^2 + f_s(P_j), \tag{3.101}$$

kde e_i je chyba aproximace a $f_s(P_j), j = 0, 1, 2, \ldots, n$ je vyhlazovací funkce (může mít různá vyjádření podle užité metody, nebo není použita vůbec). Cílovou funkci f minimalizujeme iterativně, dokud je splněna podmínka aproximace, tedy dokud $\max_i (|e_i|) \leq E$ (chyba e_i je většinou orientovaná vzdálenost, tedy může být i záporná – proto při hledání největší chyby používáme absolutní hodnoty).

Chybu aproximace e_i mezi $C(\bar{u}_i)$ a bodem Q_i můžeme určit několika způsoby:

Eukleidovská vzdálenost – point distance – PD – $e_{PD,i}$

$$e_{PD,i} := \sqrt{(Q_i - C(\bar{u}_i))^2} = \sqrt{d_i^2} = ||d_i||, i = 0, 1, 2, \ldots, m \tag{3.102}$$

Metodu minimalizace cílové funkce pomocí této vzdálenosti publikoval v roce 1988 Josef Hoschek (více v [26]) a jedná se o často používanou metodou, která se však vyznačuje pomalou konvergencí k požadované chybě aproximace.

Vzdálenost měřená od tečny – Tangent Distance – TD – $e_{TD,i}$

$$e_{TD,i} := d_i^T \cdot n_i, i = 0, 1, 2, \ldots, m, \tag{3.103}$$

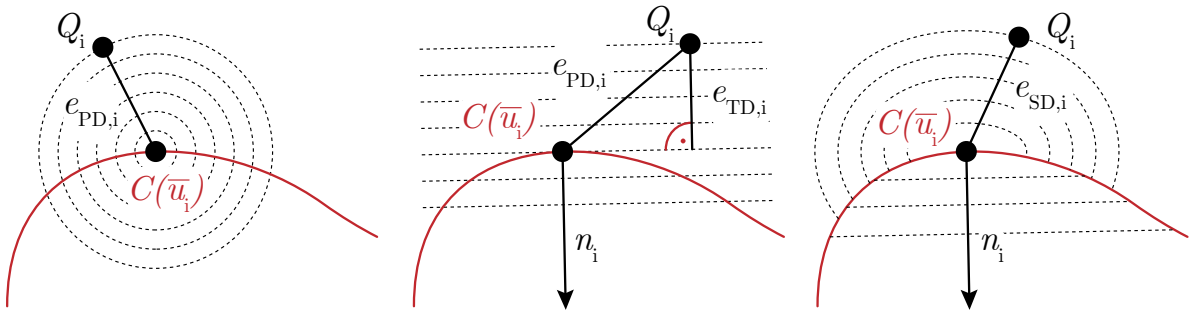
kde n_i je jednotkový vektor křivky C v bodě $C(\bar{u}_i)$ a $d_i = e_{PD,i}$ z předchozí metody. Jde o vzdálenost bodu Q_i od jeho pravoúhlého průmětu na tečnu křivky C v bodě $C(\bar{u}_k)$ (vzdálenost d_i se tedy promítne ve směru vektoru normály n_i na tečnu – viz obr. 3.39). Princip minimalizace za použití této vzdálenosti představili Andrew Blake a Michael Isard v roce 1998 [10]. Metoda konverguje rychleji než PD, ale v místech vysoké křivosti může být $e_{TD,i} = 0$, což vede k nevhodné aproximaci bodů v těchto místech.

Squared Distance – SD

$$e_{SD,i}^2 := \begin{cases} \frac{e_{TD,i}}{e_{TD,i} - \rho_i}(d_i^T t_i)^2 + (d_i^T n_i)^2 & \text{pro } e_{TD,i} < 0 \\ & i = 0, 1, 2 \ldots, m, \\ e_{TD,i}^2 & \text{pro } 0 \leq e_{TD,i} < \rho_i, \end{cases} \tag{3.104}$$

kde t_i je jednotkový tečný vektor křivky C v bodě $C(\bar{u}_i)$, tedy $t_i = C'(\bar{u}_i)$, ρ_i je poloměr křivosti křivky C v bodě $C(\bar{u}_i)$, tedy $\rho_i = ||C''(\bar{u}_i)||$. Wenping Wang *et al.* tento přístup navrhli a publikovali v roce 2006 [64]. Kombinuje PD a TD, přičemž odstraňuje jejich nedostatky a konverguje ze všech těchto metod nejrychleji.

Upravenou formu metody SD používám v navrženém algoritmu prokladu křivkou (viz kap. 4.4 str. 81).



(a) (b) (c)

Obrázek 3.39: Ilustrace vzdálenosti e_i (a) PD s izočarami pro $e_{PD,i}$, (b) TD s izočarami pro $e_{TD,i}$, (c) SD s izočarami pro $e_{SD,i}$.

3.4.6. Proklad šroubovice

Ve strojním inženýrství je šroubovice jednou z často zastoupených křivek. Především se vyskytuje na závitech šroubů a matic, dále na řezných plochách fréz či vrtáků apod. Při rekonstrukci skenovaného objektu se na těchto částech mohou vyskytnout chyby. Proto je vhodné tyto body detekovat a k jejich zobrazení použít přímo šroubovice.

V této části je matematicky popsána šroubovice a jedna z metod jejího prokládání v mračnu bodů. Většina algoritmů pro určení parametrů šroubovice ze zadaných dat mají některé parametry předdefinované a ostatní určují. S metodami se můžeme setkat například při analýze proteinových struktur v biologii; při popisu trajektorií bodů v magnetických polí v oblasti jaderné fyziky [22]; nebo v inženýrství při hledání bodů na šroubovitých tvarech strojních součástí.

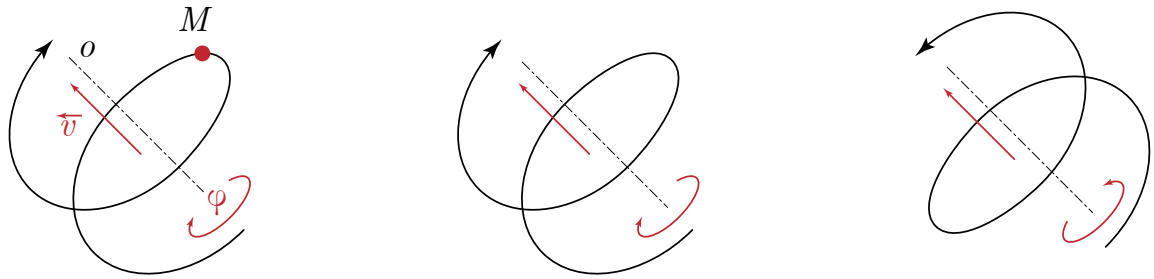
Ostatní algoritmy se většinou zaměřují pouze na určení osy šroubovice. Jde například o implementaci parametrické metody nejmenších čtverců (*parlsq*) [31], momentové matice (*eigenfit*) [32], nebo prokládání válcové plochy (*rotfit*) [40].

Enkhbayar *et al.* [18] představil v roce 2008 novou metodu *HELFIT*, která určí všechny parametry šroubovice současně pomocí metody nejmenších čtverců.

Matematický popis šroubovice

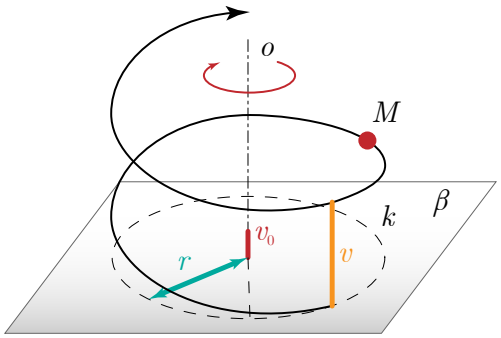
Šroubovice je křivka, která vzniká jako trajektorie bodu vykonávající tzv. *šroubový pohyb*. Ten vzniká složením posunutí a rotace, přičemž posunutí $T_{\vec{v}}$ je dáno vektorem $\vec{v}$, rotace R_φ je dána orientovaným úhlem φ a osou určenou vektorem $\vec{v}$ (vektor posunutí $T_{\vec{v}}$). Libovolný bod trajektorie je určen úhlem otočení φ a vektorem posunutí $\vec{v}$. V případě kladně orientovaného úhlu vzniká *pravotočivá* a v opačném případě *levotočivá* šroubovice (viz obr. 3.40).

Šroubovice je určena těmito parametry: poloměr r, osa o, stoupání či redukovaná výška závitu v_0, která odpovídá posunutí při otočení o jeden radián, a výška jednoho závitu šroubovice $v = 2\pi v_0$ (obr. 3.41). Na tomto obrázku je také zřejmé, že průmět šroubovice do roviny kolmé k její ose je kružnice k se stejným poloměrem.



(a) (b) (c)

Obrázek 3.40: (a) šroubový pohyb, (b) levotočivá šroubovice, (c) pravotočivá šroubovice.



Obrázek 3.41: Popis levotočivé šroubovice s průmětem do roviny β.

Parametricky lze jeden závit pravotočivé šroubovice s osou z vyjádřit následovně:

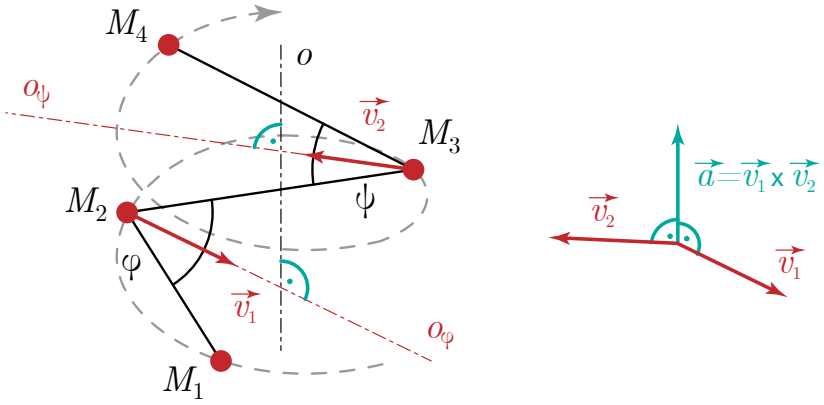
$$\begin{aligned} x &= r\cos(\varphi) \\ y &= r\sin(\varphi),\ \ \varphi \in \langle 0, 2\pi \rangle. \\ z &= v_0\varphi \end{aligned} \tag{3.105}$$

Algoritmus HELFIT

HELFIT [18] je konkrétní aplikací křivkové aproximace metodou nejmenších čtverců a pracuje v těchto třech krocích: (i) nalezení osy a poloměru vepsané válcové plochy; (ii) určení výšky závitu šroubovice; (iii) určení rovnice šroubovice. Byl specificky vyvinut pro rekonstrukci proteinových struktur, které mají šroubovitý tvar.

Osa a poloměr šroubovice

Osa (určená jednotkovým směrovým vektorem $\vec{a}$) a poloměr šroubovice r se určí pomocí vepsané válcové plochy. Její osa byla určena pomocí Kahnovi metody [32], která pracuje s postupy vektorové algebry a vystačí si se čtyřmi vstupními body. Vyberou se tři po sobě jdoucí body, ekvidistantně vzdálené od sebe; rozpůlí se jejich úhel a tím vznikne vektor $\vec{v_1}$, který bude kolmý k hledané ose. Tento postup se opakuje pro další tři body a vznikne vektor $\vec{v_2}$, který je opět kolmý k ose hledané šroubovice. Vektory $\vec{v_1}$ a $\vec{v_2}$ jsou oba kolmé na tuto osu, a proto jejich vektorovým součinem získáme její směrový vektor $\vec{a}$ (viz obr. 3.42). Tento vektor se spočítá jako jednotkový. Poloměr r byl následně určen tak, že se body promítly na rovinu kolmou k ose, a těmito průměty se proložila kružnice.



Obrázek 3.42: Ilustrační obrázek principu určení směrového vektoru osy šroubovice.

Výška závitu šroubovice

Pro určení výšky závitu P šroubovice se použije procedury nejmenších čtverců, která určí její parametr v_0, tedy lineární závislost mezi posunutím bodů a jejich otočením kolem osy šroubovice. Pomocí parametru se následně určí celková výška šroubovice.

$$P = 2\pi v_0 \tag{3.106}$$

Rovnice šroubovice

Bod X^h na šroubovici je určen touto rovnicí:

$$X^h = \vec{o} + \vec{a}Pt + r(\vec{v}\cos(t) + \vec{w}\sin(t)), \tag{3.107}$$

kde $\vec{o}$ je vektor z počátku kolmý na osu šroubovice; $\vec{v}$ je jednotkový vektor kolmý na vektor $\vec{a}$; $\vec{w}$ je jednotkový vektor kolmý na vektory $\vec{a}$ a $\vec{v}$ a t je nezávislá proměnná. Vektory $\vec{v}$ a $\vec{w}$ umožňují upravení tvaru šroubovice z kruhové na eliptickou podle potřeby. Výchozí parametry P, r, $\vec{a}$ a $\vec{o}$ této rovnice jsou již spočteny z předchozího kroku. Nyní je na řadě zjištění hodnot těchto parametrů vzhledem k celému mračnu za využití metody nejmenších čtverců.

Nechť $X_i, i = 1, 2, \ldots, n$ je bod v mračnu bodů. I-tý bod X_i^h hledané šroubovice zjistíme minimalizací jeho vzdálenosti

$$|x_i - x_i^h| := d(x_i) := d_i^h \tag{3.108}$$

od nejbližšího bodu X_i. Nechť J^h je čtverec vzdáleností bodu X_i od šroubovice.

$$J^h(P, r, \vec{a}, \vec{o}, t_0) = \sum [d(x_i)]^2, \tag{3.109}$$

kde t_0 je parametr t pro první zpracovávaný bod. Hodnotu J^h minimalizujeme vzhledem k ortogonalitě vektorů $\vec{a}$ a $\vec{o}$ (tedy $\vec{a}\cdot\vec{o} = 0$) a k jednotkové velikosti vektoru $\vec{a}$. Minimalizací postupně získáme hledané parametry: P, r, $\vec{a}$, $\vec{o}$ a t_0. Parametry $\vec{v}$ a $\vec{w}$ z rovnice šroubovice jsou volitelné.

Tento algoritmus poskytuje robustní a přesnou metodu pro prokládání mračna bodů šroubovicí, která není citlivá na šum.

4. Výsledky práce

V této kapitole je popsán nově navržený algoritmus vyhledávání bodů na specifických rysech (viz kap. 4.3 str. 66) a také vylepšený algoritmus prokladu b-spline křivky zadanými body (viz kap. 4.4 str. 81). Kapitola dále obsahuje přehled současných metod z této problematiky a metodiku výzkumu disertační práce.

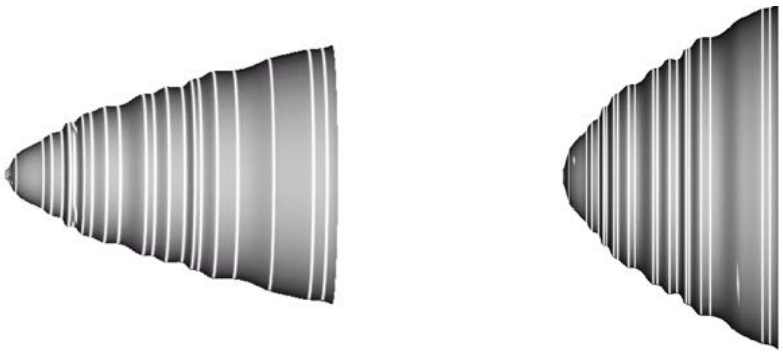
4.1. Současný stav problematiky

4.1.1. Detekce bodů na specifických rysech

Na přelomu druhého tisíciletí dochází k velkému rozmachu v oblasti 3D skenování a s tím souvisejících mračen bodů. Získanou digitální podobu snímaných objektů je následně nutné rekonstruovat a vizualizovat prostřednictvím výpočetní techniky. S metodami objektové rekonstrukce se můžeme setkat například v oblasti reverzního inženýrství či rapid prototyping. Tato odvětví využívají standardní metody zpracování mračen bodů, které však v místech ostrých hran a rohů – tzv. *sharp feature* vykazují znatelné chyby a vzniklé modely je nutné v těchto místech ručně opravovat. Je proto žádoucí takové body automaticky vyhledat a zpracovat.

Problematice vyhledávání těchto bodů se v současné době věnuje řada vědeckých prací. V následujícím textu jsou v chronologickém pořadí popsány různé metody.

Stylianou G. a Farin G. v roce 2003 publikovali metodu *Crest lines extraction from 3D triangulated meshes* [58]. *Crest line* (obr. 4.1) je lokální místo objektu s největší hlavní křivostí v příslušném hlavním směru. Tato místa si můžeme představit jako rovníkové, hrdelní či kráterové rovnoběžky rotačních ploch. Mají své využití například v medicíně v oblasti neurologie, kdy představují záhyby mozkových laloků. V navržené metodě jsou tato místa vyhledávána v triangulační mřížce vstupního mračna bodů pomocí prokladu polynomu druhého stupně metodou nejmenších čtverců.

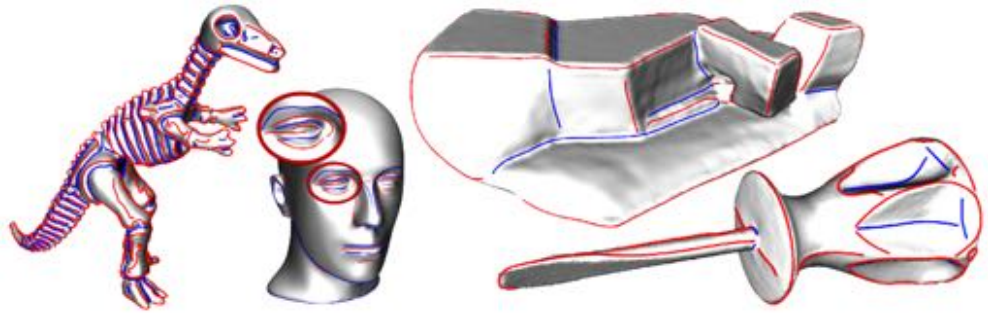


(a) (b)

Obrázek 4.1: Nalezené crest lines modelu vázy. Obrázek převzat z [58].

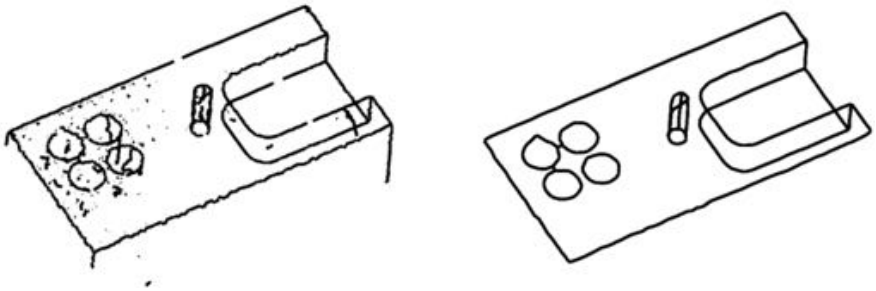
Hildebrandt K. *et al.* v roce 2005 představil metodu *Smooth Feature Lines on Surface Meshes* [25]. Metoda vyhledává ostré hrany (obr. 4.2) v triangulační mřížce jako místa lokálního maxima hlavní křivosti v příslušném hlavním směru. Metoda je v teoretickém základu podobná jako v předchozím přístupu. Liší se v samotném matematickém popisu, pomocí kterého se tyto hrany určují. V tomto případě se ostré hrany určují využitím metod

diskrétní tenzorové diferenciální geometrie. Protože algoritmus počítá se třetí derivací, jejíž výsledky jsou náchylné na šum, nalezené hrany jsou proto následně vyhlazovány.



Obrázek 4.2: Detekované ostré hrany testovaných objektů. Obrázek převzat z [25].

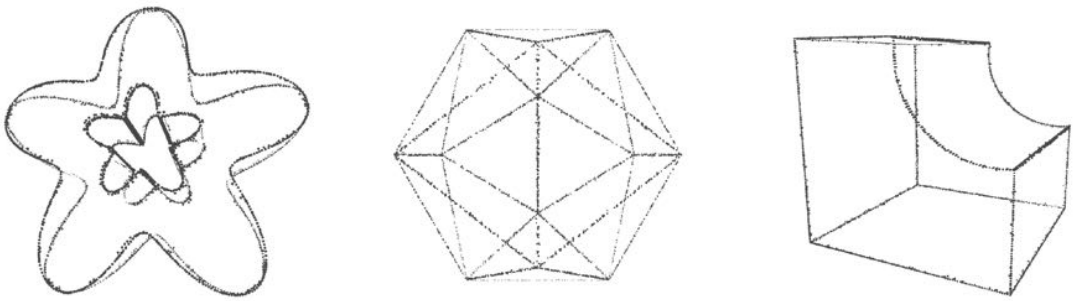
Demaris K. *et al.* publikoval v roce 2007 metodu *Detection of closed sharp edges in point clouds using normal estimation and graph theory* [15]. Oproti předchozím dvěma metodám není tento přístup závislý na předem vytvořené triangulační mřížce mračna bodů. Algoritmus vyhledává uzavřené ostré hrany (obr. 4.3) prostřednictvím teorie grafů. Nejdříve se pomocí region growing segmentace odstraní rovinné body, následně se vytvoří graf zbylých bodů a pomocí metody nejmenší kostry se vyhledají body na ostrých hranách. Nalezené hrany jsou následně vyhlazeny.



Obrázek 4.3: Detekované body a sestrojené ostré hrany testovaného objektu. Obrázek převzat z [15].

Weber C. *et al.* v roce 2010 publikoval metodu *Methods for Feature Detection in Point Clouds* [65]. Metoda pracuje přímo s mračnem bodů a nepotřebuje vstupní triangulační mřížku. V prvním kroku navržený algoritmus odstraní rovinné body pomocí region growing segmentace. Ve druhém kroku se provádí detekce bodů ostrých hran (obr. 4.4) využitím Gaussova zobrazení [6] množiny trojúhelníků vytvořené ze vstupních dat.

Xu J. *et al.* představil v roce 2015 novou metodu *Robust surface segmentation and edge feature lines extraction from fractured fragments of relics* [67]. Metoda byla vytvořena pro nalezení zlomů na fragmentech archeologických nálezů pro jejich následnou rekonstrukci. Algoritmus vyhledává body ostrých hran pomocí region growing segmentace založené na změně normálových vektorů v mračnu bodů. V prvním kroku dochází k filtrování vstupních dat pro odstranění šumu. Ve druhém kroku dojde k hrubé segmentaci na velký počet jednotlivých části s malou změnou odchylky normál. Třetí krok provádí jemnější region growing segmentaci, ve které se sjednocují clustery nalezené v předchozím kroku do celkových ploch tělesa. V posledním kroku dochází k samotné detekci bodů ostrých hran. Provádí se zde propojování dosud nespojených clusterů do odpovídajících částí s malou změnou normál. Body na hranici mezi těmito clustery jsou hledané body zlomu (obr. 4.5).



(a) (b) (c)

Obrázek 4.4: Detekované body ostrých hran na vstupních modelech.

(a) (b)

Obrázek 4.5: Detekované body zlomu na vstupních fragmentech.

Algoritmus je určen výhradně pro vyhledávání bodů zlomů částí archeologických nálezů. Tyto fragmenty mají specifické tvary, které určují místa, ve kterých došlo k rozlomení původního celku na jednotlivé části.

Jak je vidět z obr. 4.5, metoda detekuje pouze body, ve kterých došlo ke zlomu původního objektu. Tento fakt je způsoben prvotní hrubou segmentací, kdy například oděrky, rýhy či jiná ostrá místa jsou zahrnuty do počátečních clusterů a nepovažují se tedy za hrany zlomu. Vše vyplývá ze zkušenosti s reálnými fragmenty, které na sebe navazují vnějšími zlomovými hranami a vytváří tak daný předmět. Při aplikaci této metody v technické praxi by některé ostré hrany byly ignorovány, což by vedlo k jejich chybnému určení.

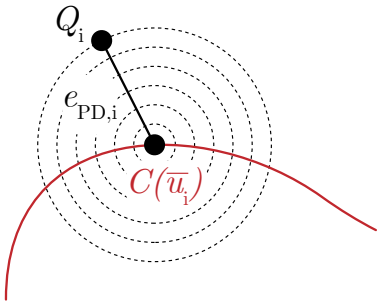
4.1.2. Proklad křivkou

Další část pojednává o současných metodách prokladu křivkou. Nejčastěji používanou křivkou pro prokládání je b-spline či racionální b-spline, tedy NURBS. Metody prokladu touto křivkou se liší způsobem, kterým určují chybu její aproximace, tedy vzdálenost bodů křivky od vstupních dat. Zde jsou v chronologickém uspořádání popsány vybrané metody prokladu křivkou.

Hoschek J. publikoval v roce 1988 metodu prokladu parametrických křivek a ploch v publikaci *Intrinsic parametrization for approximation* [26]. Metody aproximace jsou založeny na minimalizaci vzdálenosti, která se určuje klasicky jako eukleidovská vzdálenost

dvou bodů (obr. 4.6). Tato vzdálenost je zde označena jako *point distance – PD*. Chyba aproximace bodu Q_i od křivky $C(u_i)$ určovaná tímto přístupem, $e_{PD,i}$ je v tomto tvaru:

$$e_{PD,i} := \sqrt{(Q_i - C(u_i))^2} = \sqrt{d_i^2} = ||d_i||, i = 0, 1, 2, \ldots, m. \tag{4.1}$$



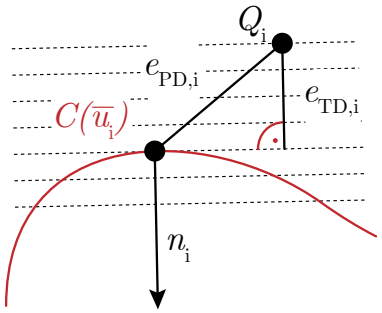
Obrázek 4.6: Ilustrace vzdálenosti PD s izočarami.

Metoda aproximace křivky minimalizací této vzdálenosti se zde následně označuje jako *point distance minimalization – PDM*. Tato metoda je velmi intuitivní, robustní a má jednoduchou implementaci ve výpočetní technice. Ovšem nevýhodou je pomalá konvergence k požadovanému řešení.

O deset let později, v roce 1998 publikovali Blake A. a Issard M. metodu minimalizace za využití tzv. *tangent distance – TD*. Tato vzdálenost, *měřená od tečny*, je v daném bodě Q_i sestrojena jako průmět jeho PD vzdálenosti ve směru normály hledané křivky v bodě $C(u_i)$ (odpovídající bodu Q_i) na tečnu v bodě $C(u_i)$ (viz obr. 4.7). Tato vzdálenost e_{TD} má tvar

$$e_{TD,i} := d_i^T \cdot n_i, i = 0, 1, 2, \ldots, m, \tag{4.2}$$

kde n_i je jednotkový normálový vektor křivky v bodě $C(u_i)$.



Obrázek 4.7: Ilustrace vzdálenosti TD s izočarami.

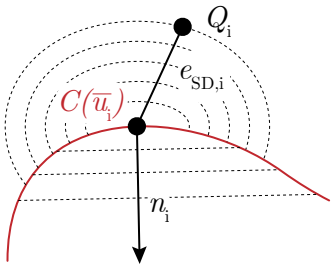
Metoda prokladu křivkou tímto způsobem se zde označuje *tanget distance minimalizatioin – TDM*. Vzdálenost TD může být nulová, pokud se pod Q_i nachází blízko, případně přímo na tečně hledané křivky v bodě $C(u_i)$, což může nastat v místě velké křivosti. V těchto místech tedy metoda TDM špatně aproximuje tvar vstupních bodů. Metoda TDM konverguje rychleji než PDM, ale v místech vysoké křivosti vykazuje nestabilní výsledky.

Wang W. *et al.* v roce 2006 představil v publikaci *Fitting B-Spline Curves to Point Clouds by Squared Distance Minimization* [64] metodu kombinující robustnost PDM

a rychlou konvergenci TDM. Zavedl princip určování chyby aproximace metodou tzv. *squared distance – SD* (*čtvercová vzdálenost*) (obr. 4.8), která má v bodě Q_i vyjádření

$$e^2_{SD,i} := \begin{cases} \frac{e_{TD,i}}{e_{TD,i}-\rho_i}(d_i^T t_i)^2 + (d_i^T n_i)^2 & \text{pro } e_{TD,i} < 0 \\ & i = 0, 1, 2 \ldots, m, \\ e^2_{TD,i} & \text{pro } 0 \leq e_{TD,i} < \rho_i, \end{cases} \tag{4.3}$$

kde t_i je jednotkový tečný vektor křivky C v bodě $C(u_i)$, tedy $t_i = C'(u_i)$, ρ_i je poloměr křivosti křivky C v bodě $C(u_i)$, tedy $\rho_i = ||C''(u_i)||$.



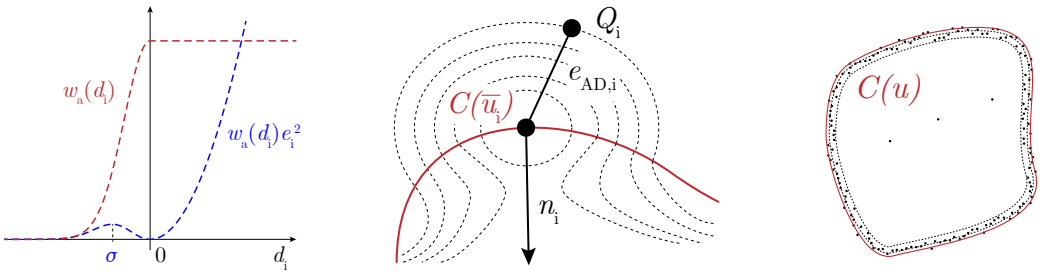
Obrázek 4.8: Ilustrace vzdálenosti SD s izočarami.

Metoda minimalizace s touto chybou se označuje jako *squared distance minimalization – SDM* a z těchto tří metod konverguje nejrychleji a poskytuje konzistentní výsledky.

Mörwald T. *et al.* v roce 2016 publikoval metodu *Modeling connected regions in arbitrary planar point clouds by robust B-spline approximation*, která je upravenou verzí SDM. V tomto přístupu zavádí váženou SD vzdálenost označenou jako *asymetric distance – AD* (*asymetrická vzdálenost*) a má pro bod Q_i tento tvar

$$e^2_{AD_i} = w_a(d_i)e_i^2, \qquad \text{kde } w_a(d_i) := \begin{cases} e^{-\frac{d_i^2}{\sigma^2}} & \text{pro } d_i < 0 \\ & i = 0, 1, 2 \ldots, m, \\ 1 & \text{pro } d_i \geq 0 \end{cases} \tag{4.4}$$

kde $e_i^2 = e^2_{SD,i}$ a $\sigma \in \mathbb{R}^+$ je hodnota posunu váhové funkce vzhledem k orientované vzdálenosti $d_i = e_{TD,i}$ (obr. 4.9(a)). Za použití této váhy se křivka v iteracích přibližuje více k bodům na okraji vstupní množiny (obr. 4.9(c)).



(a) (b) (c)

Obrázek 4.9: (a) graf váhy a vzdálenosti, (b) ilustrace vzdálenosti AD, (c) křivka.

4.2. Metodika

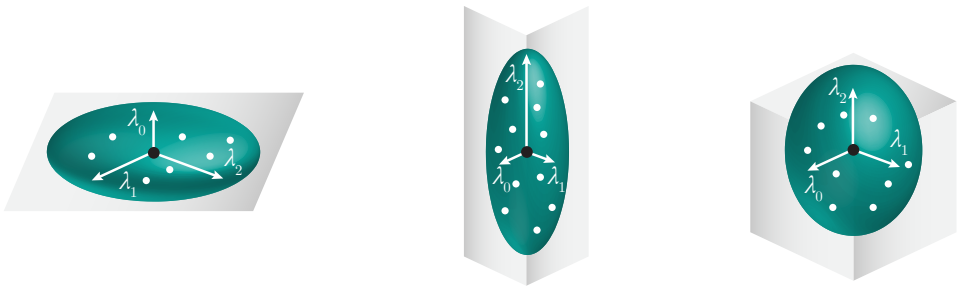
V této části je uveden metodický postup při vytváření nové metody vyhledávání bodů na specifických rysech v mračnu bodů i při vytváření metody jejich prokladu křivkou.

4.2.1. Detekce bodů na specifických rysech

Na základě nastudování současné literatury k tématu vyhledávání bodů na specifických rysech v mračnu bodů jsem se zajímal o tyto používané metody:

- *PCA – principal component analysis* s českým ekvivalentem *analýza hlavních komponent* je jednou z často používaných metod detekce bodů na specifických rysech v mračnu bodů. Je založena na výpočtu kovarianční matice v okolí zkoumaného bodu. Tato matice popisuje rozptyl okolních bodů vzhledem k jejich těžišti. Matice je reálná, pozitivně semidefinitní a symetrická. Výpočtem vlastních čísel $(\lambda_0, \lambda_1, \lambda_2)$ a příslušných vlastních vektorů $(\vec{v_1}, \vec{v_2}, \vec{v_3})$ této matice je možné rozhodnout o tvaru okolí zkoumaného bodu. Protože je kovarianční matice symetrická, má právě tři různá reálná vlastní čísla, kterým odpovídají tři různé navzájem kolmé vlastní vektory (důkaz je možné najít například v [51]). Vlastní vektory určují směry os (vlastní čísla jejich velikost) tzv. *kovariančního elipsoidu*, jehož tvar popisuje okolí daného bodu. Pro velikosti vlastních čísel, tedy i tvar kovariančního elipsoidu, můžou nastat tři různé případy:

  1. $(\lambda_0 \leq \lambda_1 \leq \lambda_2) \wedge (\lambda_1 \approx \lambda_2)$ V tomto případě je okolí daného bodu podobné rovině, kde λ_0 určuje nejmenší rozptyl ve směru $\vec{v_0}$, který se ztotožní s normálovým vektorem roviny proložené okolními body. Kovarianční elipsoid je tedy zploštělý (obr. 4.10(a)).
  2. $(\lambda_0 \leq \lambda_1 \leq \lambda_2) \wedge (\lambda_0 \approx \lambda_1)$ V tomto případě, kdy jsou vypočtena dvě velikostně podobná vlastní čísla výrazně menší než třetí z nich, se v okolí zpracovávaného bodu pravděpodobně setkávají dvě plochy s velkou odchylkou normál, jde tedy o body hrany. Kovarianční elipsoid má protáhlý tvar (obr. 4.10(b)).
  3. $(\lambda_0 \leq \lambda_1 \leq \lambda_2) \wedge (\lambda_0 \approx \lambda_1 \approx \lambda_2)$ Jedná se o případ tří podobně velkých vlastních čísel. Okolní body tedy leží na vícero různých plochách. Kovarianční elipsoid je tedy podobný kulové ploše (obr. 4.10(c)).



(a) (b) (c)

Obrázek 4.10: Kovarianční elipsoid (a) zploštělý, (b) protáhlý, (c) podobný kulové ploše.

Detekování touto metodou je pro mračna s velkým počtem bodů výpočetně náročné. Tímto způsobem se nejdříve z mračna odstraní rovinné body, následně se analýza provede znovu pro nalezení hledaných bodů ostrých hran. Vícenásobná analýza je výpočetně náročná, a proto se používá až pro kandidáty bodů specifických rysů určených jinou metodou.

- *Region growing segmentace* (viz kap. 3.1.3 str. 18), která mračno rozdělí na shluky podle malé změny odchylky normál, se u většiny dnešních metod používá pouze pro vyhledání a odstranění rovinných bodů. Jediná metoda používající přímo region growing segmentaci je metoda *Robust surface segmentation and edge feature lines extraction from fractured fragments of relics* [67]. Ta je ale velmi specifická a je určená pouze pro použití při vyhledání míst zlomů v nalezených fragmentech archeologických nálezů (viz str. 60).

- *Gaussovo zobrazení* (viz kap. 3.1.3 str. 22) je možné použít jak pro vyhledávání bodů na specifických rysech v mračnu bodů, tak pro jejich kategorizaci na rovinné body, body hran či body rohů podobně jako v případě PCA. Pro mračna s vysokým počtem bodů je metoda opět výpočetně náročná a je nutné několikanásobné provedení. V prvním průchodu se odstraní rovinné body, v druhém se vyberou a kategorizují hledané body.

Po prostudování těchto metod, jejich výhod a nevýhod jsem vytvořil novou metodu založenou na region growing segmentaci pro nalezení hledaných bodů na specifických rysech. Region growing segmentace je v současnosti dobře optimalizovaný algoritmus, který dokáže v relativně krátkém čase zpracovat velké množství dat. Navíc je tato optimalizovaná metoda implementována i v knihovně PCL.

Moje metoda v mračnu bodů odstraní všechny body, které se nachází v místech zkoumaného objektu, kde dochází k předem nastavené maximální odchylce normálových vektorů. Oproti zmíněným metodám můj přístup umožní pomocí region growing segmentace automatické odstranění bodů jak rovinných částí, tak i částí zakřivených. Body, které po segmentaci v mračnu zůstanou, jsou hledané body na specifických rysech – ostrých hran. Moje nově navržená metoda bude pracovat ve dvou základních krocích: (i) určení normálových vektorů v místech všech bodů v mračnu bodů užitím metody kovarianční matice (viz kap. 4.3.1 str. 67); (ii) odstranění bodů v místech s definovanou maximální odchylkou normálových vektorů pomocí region growing segmentace (viz kap. 4.3.1 str. 67).

4.2.2. Proklad křivkou

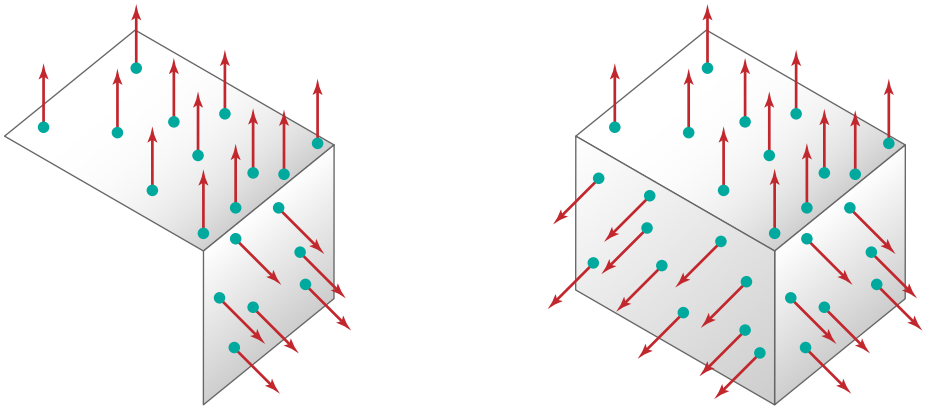
V oblasti prokladu křivkou v mračnu bodů se nejčastěji pracuje s křivkami typu b-spline (viz kap. 3.3.5), proto ji používám i ve své metodě. Metody prokladu se liší ve způsobu určování chyby aproximace v dané iteraci, kterou minimalizují. Jedná se o metody: eukleidovská vzdálenost – PD, vzdálenost měřená od tečny – TD, čtvercová vzdálenost – SD, metody jsou popsány v kap. 3.4.5.

Nejlepší z těchto metod je měření pomocí čtvercové vzdálenosti – *squared distance – SD*, která je robustní a má rychlou konvergenci. V minulosti byla upravena vhodným vážením, které způsobuje, že se prokládaná křivka přibližuje k bodům na okraji vstupní množiny. Tato upravená verze nese označení *asymetric distance* (*asymetrická vzdálenost*) – *AD*. Tuto metodu se pokusím více upravit výběrem nové vhodné váhy tak, aby se hledaná

křivka ještě více přibližovala k okrajovým bodům, což se bude více hodit k výsledkům mého navrženého algoritmu detekce bodů ostrých hran.

4.3. Detekce bodů na specifických rysech

Tato kapitola je věnována nově navrženému algoritmu pro vyhledávání bodů na specifickém rysu tzv. (*feature*) v mračnu bodů. Nalezení těchto bodů napomáhá ke zpřesnění rekonstrukce daného objektu – především v oblasti reverzního inženýrství, objektové rekonstrukce, či průmyslového designu. Feature – v našem případě uvažujeme ostrou hranu či roh, je místo, kde na sebe navazují roviny a plochy s výrazně různými normálovými vektory či křivostmi (obr. 4.11). Této hlavní myšlenky jsem využil při navržení nového algoritmu.



(a) (b)

Obrázek 4.11: Specifický rys (feature) (a) hrana, (b) roh.

4.3.1. Algoritmus detekce

Hlavní myšlenka navrženého algoritmu spočívá v tom, že body, které detekuji se nachází v místech předmětů, kde se setkávají plochy s rozdílnými normálovými vektory a křivostmi (obr. 4.11). Na okolí hledaných bodů dochází k větší změně normál a křivostí než v samotných okolních plochách. Pokud z mračna odstraním body, v jejichž okolí dochází k malé změně normál a křivostí – menší než zvolený práh, pak zbylé body jsou s vysokou pravděpodobností hledané body zájmu. K odstranění těchto bodů je využito region growing segmentace (viz kapitola 3.1.3 str. 16).

Navržený algoritmus je oproti současným metodám jedinečný svým využitím segmentace. Většina současných metod z této problematiky využívá region growing segmentaci pouze jako preproces, ve kterém odstraňují pouze rovinné body. V mém algoritmu však mohu odstranit i body zaoblených ploch, které se v technické praxi vyskytují stejně tak jako rovinné části. Segmentačním odstraňováním bodů v místech malé změny normál mohu v závislosti na prahu odchylek určovat, jak velký počet bodů na specifickém rysu chci najít.

Metoda detekce byla naprogramována v prostředí Visual Studio Community 2015, verze 14, za podpory PCL verze 1.8.1. Algoritmus se skládá z následujících částí: (i) výpočet normálových vektorů v místech všech bodů; (ii) region growing segmentace.

Výpočet normálových vektorů a křivosti

Normálový vektor a křivost jsou klíčové geometrické vlastnosti, pomocí kterých lze algebraicky popsat křivky či plochy v mračnu bodů. Tyto vlastnosti lze v mračnu bodů vypočítat v místě každého bodu pomocí statistických nebo geometrických metod.

V mračnu bodů není definována plocha ani křivka, na které by body ležely a tudíž nemůžeme pomocí klasické geometrie určit tečné a normálové vektory v těchto bodech. Jsou pouze informace o poloze jednotlivých bodů v prostoru pomocí jejich souřadnic (x, y, z). Statistickou metodou však mohu určit rozptyl k-nejbližších okolních bodů P_i zvoleného bodu P. Pokud tyto rozptyly určím v určitých směrech, zjistím tím, jak moc se okolí daného bodu P v těchto směrech mění. K tomuto výpočtu mohu využít statistickou metodu kovariance, kdy spočítám kovarianční matici k–nejbližších okolních bodů určeného bodu P. Vlastní vektory pak určí směry rozptylů a vlastní čísla jejich hodnoty.
Kovarianční matice má tento tvar:

$$C = \frac{1}{k}\sum_{i=1}^{k}(P_i - \bar{P}) \cdot (P_i - \bar{P})^T, \tag{4.5}$$

kde $P_i, i = 1, 2, \ldots k$ jsou sousední body zpracovávaného bodu P a $\bar{P}$ je jejich těžiště. Tato matice má v bodě P_i tento tvar:

$$C = \begin{pmatrix} (P_{i1} - \bar{P}_1)^2 & (P_{i1} - \bar{P}_1) \cdot (P_{i2} - \bar{P}_2) & (P_{i1} - \bar{P}_1) \cdot (P_{i3} - \bar{P}_3) \\ (P_{i2} - \bar{P}_2) \cdot (P_{i1} - \bar{P}_1) & (P_{i2} - \bar{P}_2)^2 & (P_{i2} - \bar{P}_2) \cdot (P_{i3} - \bar{P}_3) \\ (P_{i3} - \bar{P}_3) \cdot (P_{i1} - \bar{P}_1) & (P_{i3} - \bar{P}_3) \cdot (P_{i2} - \bar{P}_2) & (P_{i3} - \bar{P}_3)^2 \end{pmatrix}, \tag{4.6}$$

kde $P_i = (P_{i1}, P_{i2}, P_{i3}), \bar{P} = (\bar{P}_1, \bar{P}_2, \bar{P}_3)$. Kovarianční matice je reálná, pozitivně semidefinitní a symetrická. Z vlastnosti symetrie vyplývá, že kovarianční matice má právě tři různá reálná vlastní čísla, kterým odpovídají právě tři různé navzájem ortogonální vlastní vektory.

Následně se spočítají vlastní čísla $\lambda_0 \leq \lambda_1 \leq \lambda_2$ a vlastní vektory $\vec{v}_0, \vec{v}_1, \vec{v}_2$ matice C pomocí rovnice:

$$C \cdot \vec{v}_j = \lambda_j \cdot \vec{v}_j, j \in \{0, 1, 2\}, \text{ tedy } (C - \lambda E) \cdot \vec{v}_j = \vec{o}, \tag{4.7}$$

která má řešení, jestliže determinant matice soustavy je roven 0:

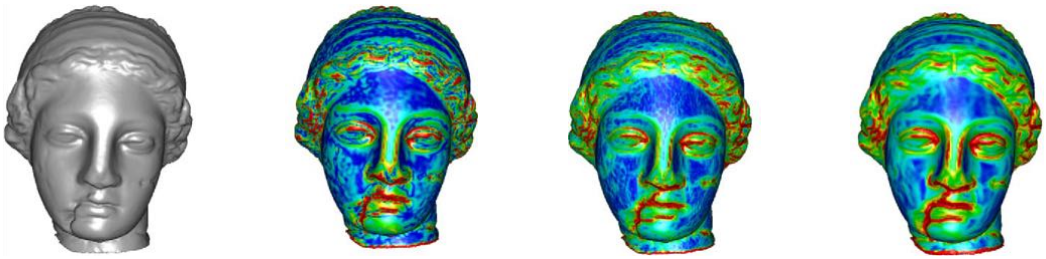
$$|C - \lambda E| = 0. \tag{4.8}$$

Vlastní vektor příslušný nejmenšímu vlastnímu číslu (λ_0) – v tomto směru je nejmenší rozptyl okolních bodů (body tedy leží na ploše podobné rovině) – ztotožním s normálovým vektorem proložené plochy v daném bodě [7]. Za pomoci vlastních čísel lze spočítat tzv. *surface variation* S_{var}, která určuje poměrný povrchový rozptyl bodů na povrchu tělesa vzhledem k nejmenší hodnotě rozptylu λ_0.

$$S_{var} = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}, \text{ kde } \lambda_0 \leq \lambda_1 \leq \lambda_2. \tag{4.9}$$

Tato hodnota poskytuje při práci s mračny bodů přesnější informaci než průměrná křivost a je používána v soudobých algoritmech jako její náhrada [47]. Hodnota S_{var}

(obr. 4.12) závisí na počtu k-nejbližších sousedních bodů, které jsou použity k určení kovarianční matice (4.5).



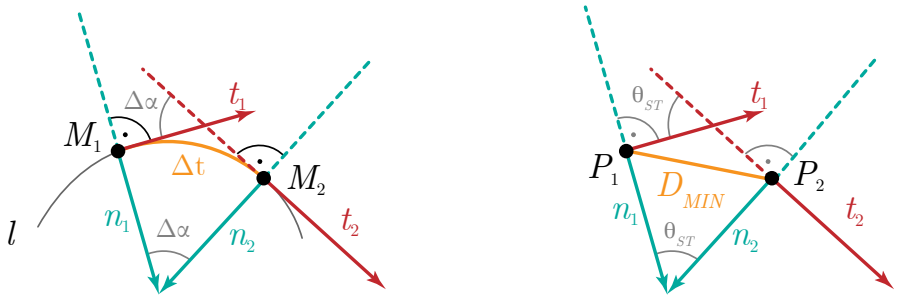
(a) (b) (c) (d)

Obrázek 4.12: (a) původní povrch, (b) průměrná křivost, (c) s_{var} ($k = 20$), (d) s_{var} ($k = 50$), obrázky převzaty z [7].

Region growing segmentace

Tato část tvoří samotné jádro navrženého algoritmu. Pomocí region growing segmentace z mračna odeberu body s maximální odchylkou normál θ_{ST} – *Surface Threshold*, ke které určím příslušný práh křivosti θ_{CT} – *Curvature Threshold*. Křivost křivky je geometricky definována jako rychlost spojité změny odchylky α tečných (případně normálových) vektorů ke spojité změně parametru t určující polohu bodu na křivce (viz obr. 4.13). Mračno bodů je oproti křivce či ploše diskrétní a tudíž mezi body mračna není definována spojitá křivka či plocha. Opět znám pouze souřadnice (x, y, z) všech bodů. Hodnotu změny odchylky normál ztotožním s hodnotou θ_{ST}. Je nutné tedy určit velikost změny polohy bodů v mračnu. Tuto informaci mohu statisticky aproximovat jako minimální vzdálenost D_{MIN} mezi body v mračnu. Práh křivosti θ_{CT} v závislosti na θ_{ST} tedy určím následovně.

$$\theta_{CT} = \frac{\theta_{ST}}{D_{MIN}} \approx \frac{\Delta\alpha}{\Delta t} \tag{4.10}$$



(a) (b)

Obrázek 4.13: (a) odvození křivosti na křivce, (b) odvození křivosti v mračnu bodů.

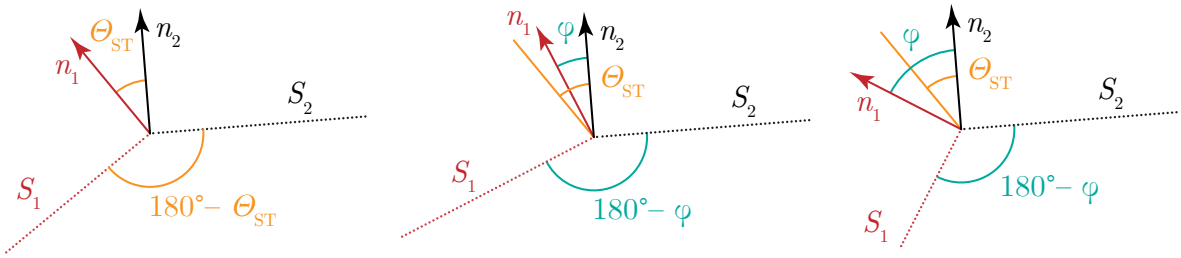
Nyní jsou tedy určené potřebné prahy pro region growing segmentaci mračna. K tomuto účelu využiji optimalizovanou funkci **RegionGrowing** z knihovny PCL, která má právě tyto hodnoty (θ_{ST} a θ_{CT}) jako vstupní parametry. V originálním kódu algoritmu této segmentace jsou obě hodnoty pevně dané a nemají mezi sebou geometrické vztahy. Proto jsem do funkce implementoval vzorec (4.10), a segmentaci řídím pouze paramet-

rem θ_{ST}, čímž jsem zaručil větší konzistenci výstupu. Algoritmus je také upraven tak, že nalezené shluky bodů automaticky z mračna odstraňuje.

4.3.2. Odhad parametru θ_{ST}

Prvotní verze algoritmu pro detekci bodů na ostrých hranách byla řízena manuálně nastavitelnou hodnotou parametru θ_{ST}. Cílem je určit takovou hodnotu, aby výsledné mračno po odebrání vysegmentovaných bodů obsahovalo přiměřené množství bodů specifických rysů pro jejich správnou vizualizaci. Během segmentací provedených na testovacích modelech jsem empiricky určil, že ideální poměr nalezených bodů $NOPO$ (*Number Of Point Out*) ku celkovému počtu bodů vstupního mračna $NOPI$ (*Number Of Point In*) je v rozmezí 3–6%. Hodnotu tohoto poměru jsem označil jako $PERC = \frac{NOPO}{NOPI} \cdot 100 \in \langle 3, 6 \rangle$. Hledám tedy takový parametr θ_{ST}, aby hodnota $PERC$ byla v rozmezí $\langle 3, 6 \rangle$.

Odchylka vektorů normál může být z geometrického hlediska až 180°, tedy parametr θ_{ST} může nabývat hodnot $(0, 180)$. Pro velké hodnoty prahu θ_{ST} se při segmentaci mohou odstranit až všechny body mračna. Naopak při velmi malé hodnotě blízké nule se nevysegmentují téměř žádné body. Proto jsem θ_{ST} omezil na hodnoty $0.5 < \theta_{ST} \leq 15$, které reflektují lokální změny normál na testovaných reálných objektech. Při hodnotě prahu $\theta_{ST} = 15$ se odstraní body ploch s lokální změnou normál φ do této hodnoty a zůstanou body ploch s vyšší lokální změnou. Touto metodou je tedy možné detekovat body ostrých hran, kde hrana má maximální úhel $180° - 15° = 165°$, což pokryje celou škálu těles různých tvarů. Princip je ilustrován na obr. 4.14.



(a) (b) (c)

Obrázek 4.14: Znázornění významu θ_{ST} na okolí ostré hrany: (a) geometrický význam, (b) $\varphi < \theta_{ST}$ – body se odstraní, (c) $\varphi > \theta_{ST}$ – body zůstanou.

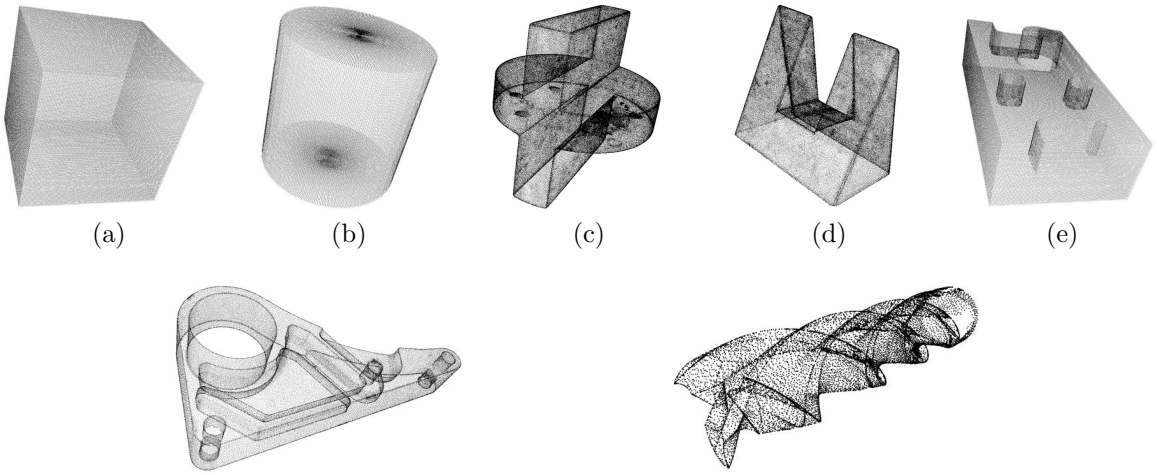
S určenou maximální hodnotou θ_{ST} jsem pak pro odvození tohoto parametru postupoval následovně:

- provedení upravené region growing segmentace pro testovací modely s různými hodnotami $\theta_{ST} = \{0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$,
- výpočet $PERC$ – procentuální zastoupení nalezených bodů specifického rysu vzhledem k celkovému počtu bodů v mračnu
- zakreslení hodnot θ_{ST} a $PERC$ do grafu,
- odvození průběhu závislosti θ_{ST} na $PERC$ pomocí regresní metody,
- porovnání regresní funkce mezi modely,
- vytvoření průměrného regresního modelu.

Mračna bodů pro odvození parametru θ_{ST}

Testy pro odvození θ_{ST} byly provedeny na mračnech, které jsem měl k dispozici. Tato mračna jsou dvojího typu: (i) data získaná využitím fakultního skeneru ATOS Compact-Scan 2M, které jsem naskenoval; (ii) data vytvořená v 3D editoru Blender. Veškerá data uložena ve formátu PCD (viz kapitola 3.1.2 str. 10).

Měl jsme k dispozici 3D skeny dřevěných modelů – model 1 a model 2, určených pro výuku deskriptivní geometrie, které byly zapůjčeny z Katedry Algebry a Geometrie Univerzity Palackého v Olomouci. Tyto modely byly naskenovány bez spodních částí. Dále sken konzole a frézy. Ostatní modely – krychle, válec a kompozit byly vytvořeny uměle v programu Blender. Modely jsou zobrazeny na obr. 4.15 v 3-úběžníkové perspektivě; tabulka 4.1 obsahuje informace o počtu bodů.



(a) (b) (c) (d) (e) (f) (g)

Obrázek 4.15: Zobrazení testovacích mračen bodů: (a) krychle, (b) válec, (c) model 1, (d) model 2, (e) kompozit, (f) konzole, (g) fréza.

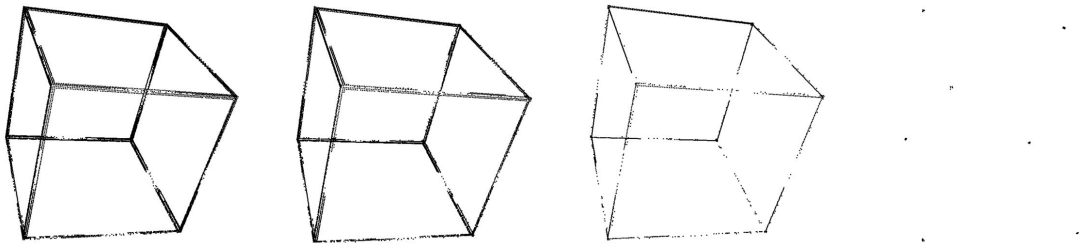
Tabulka 4.1: Vstupní mračna s příslušným počtem bodů NOP (*number of points*).

model	krychle	válec	model 1	model 2	kompozit	konzole	fréza
NOP	87 848	67 394	195 815	123 112	103 534	90 209	15 360

Výsledky algoritmu detekce na testovacích mračnech

V této části jsou v prostředí **PCLVisualizer** znázorněny výsledky detekce bodů ostrých hran, které slouží pro odhad závislosti θ_{ST} na $PERC$. Testovány byly všechny modely kromě frézy, kterou jsem z důvodu nedostatečného množství bodů nepovažoval za vhodný vzorek. Vždy je zobrazen výsledek algoritmu pro různé hodnoty θ_{ST}; tabulka výsledků $PERC$ a příslušné hodnoty θ_{ST}; graf závislosti θ_{ST} na $PERC$ včetně statistické regrese; a porovnání vstupu s výsledkem pro předurčenou hodnotu $\theta_{ST} = 3.28$ a s hodnotou θ_{ST} pro optimální výsledek nalezení 3–6% bodů. Předurčená hodnota θ_{ST} je vypočítána z konečné regresní funkce pro požadované procento 3.1% nalezených bodů (podrobné odvození na str. 78).

Nalezené body modelu krychle

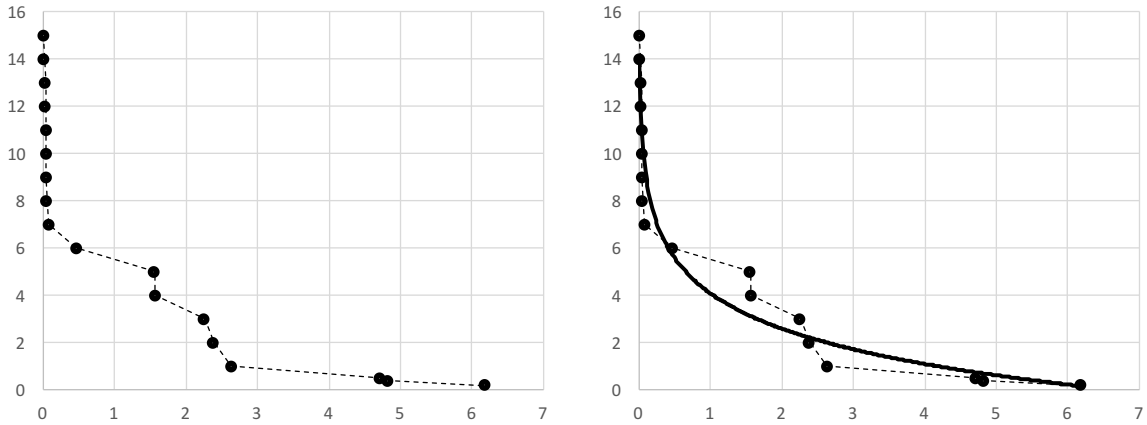


(a) (b) (c) (d)

Obrázek 4.16: Krychle: (a) $\theta_{ST} = 0.2$, (b) $\theta_{ST} = 0.5$, (c) $\theta_{ST} = 5$, (d) $\theta_{ST} = 9$.

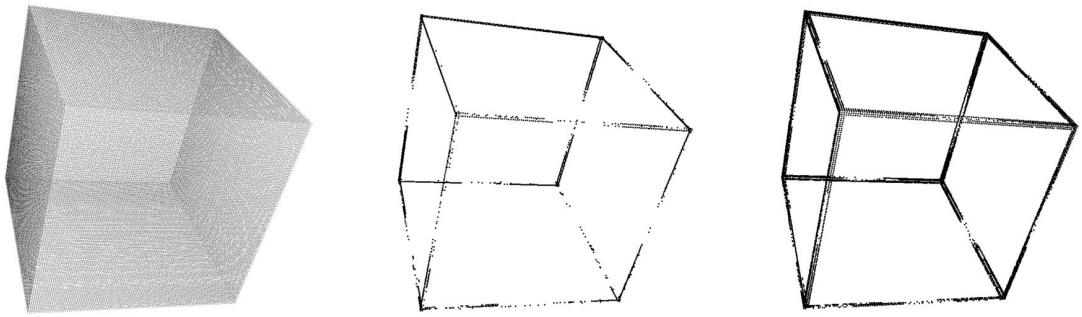
Tabulka 4.2: Tabuka θ_{ST} a $PERC$ pro mračno krychle.

θ_{ST}	0.5	1	2	3	4	5	6	7	8–11	12–13	14	15
$PERC$	4.71	2.64	2.39	2.26	1.57	1.56	0.47	0.08	0.05	0.04	0.02	0.01



(a) (b)

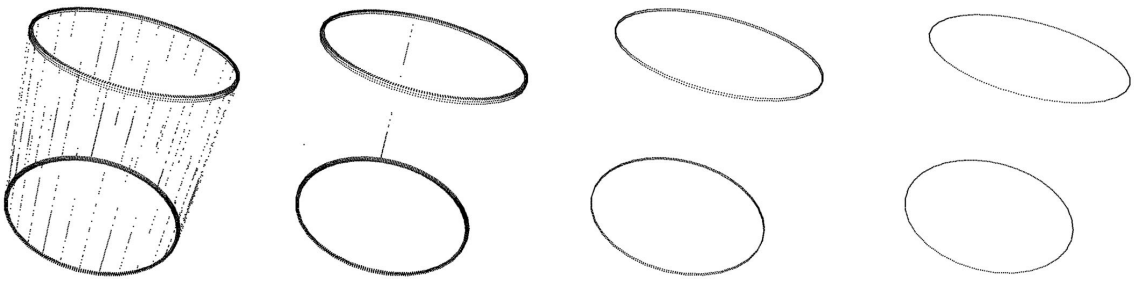
Obrázek 4.17: Krychle: (a) graf dat tabulky 4.2, (b) logaritmická regrese dat.



(a) (b) (c)

Obrázek 4.18: Krychle: (a) vstup, (b) $\theta_{ST} = 3.28$, (c) 6.18% pro $\theta_{ST} = 0.2$.

Nalezené body modelu válce

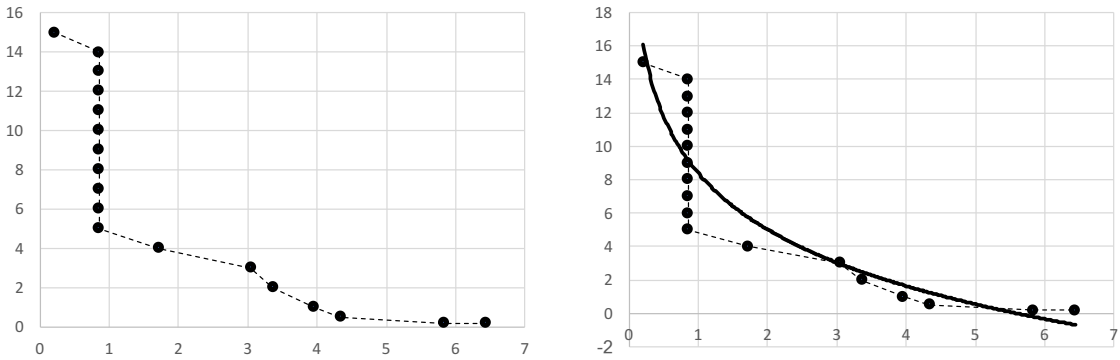


(a) (b) (c) (d)

Obrázek 4.19: Válec: (a) $\theta_{ST} = 0.2$, (b) $\theta_{ST} = 0.5$, (c) $\theta_{ST} = 4$, (d) $\theta_{ST} = 13$.

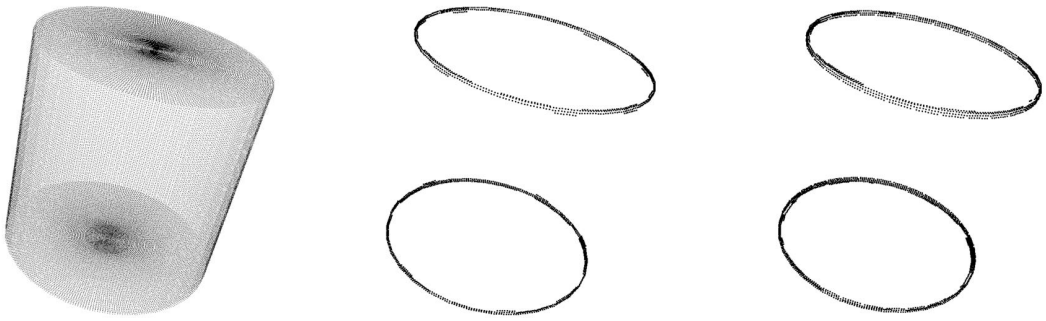
Tabulka 4.3: Tabuka θ_{ST} a $PERC$ pro mračno válce.

θ_{ST}	0.5	1	2	3	4	5–14	15
$PERC$	4.34	3.96	3.36	3.05	1.71	0.85	0.2



(a) (b)

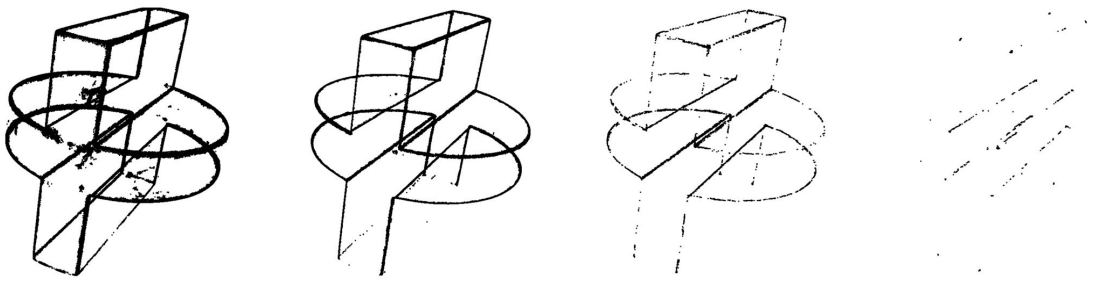
Obrázek 4.20: Válec: (a) graf dat tabulky 4.3, (b) logaritmická regrese dat.



(a) (b) (c)

Obrázek 4.21: Válec: (a) vstup, (b) $\theta_{ST} = 3.28$, (c) 3.36% pro $\theta_{ST} = 2$.

Nalezené body modelu 1

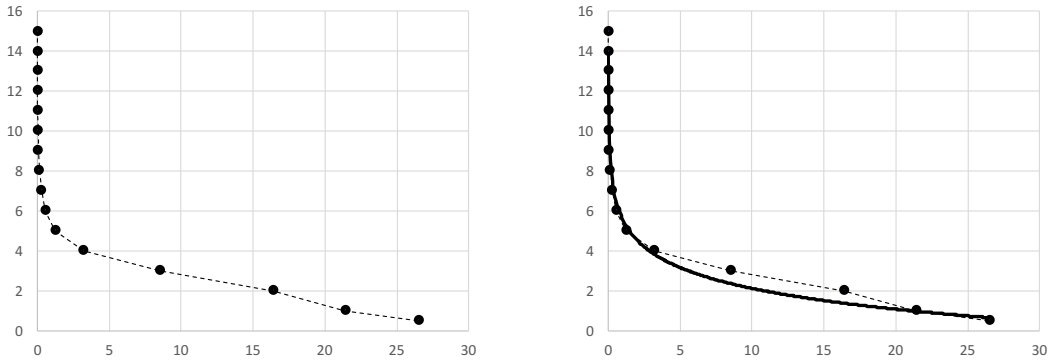


(a) (b) (c) (d)

Obrázek 4.22: Model 1: (a) $\theta_{ST} = 0.5$, (b) $\theta_{ST} = 2$, (c) $\theta_{ST} = 4$, (d) $\theta_{ST} = 7$.

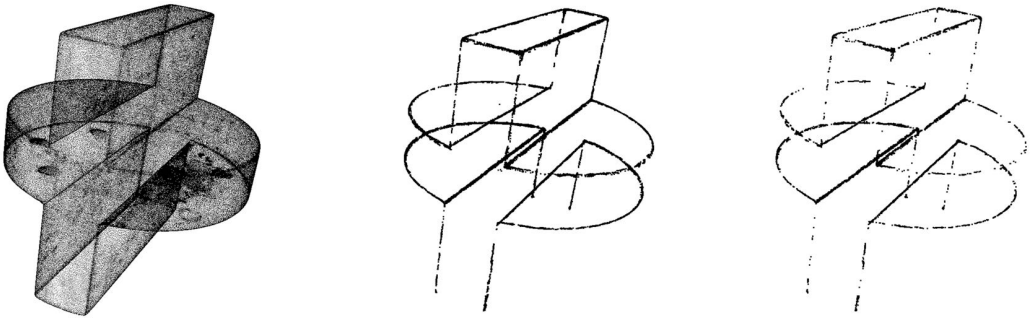
Tabulka 4.4: Tabuka θ_{ST} a $PERC$ pro mračno modelu 1.

θ_{ST}	0.5	1	2	3	4	5	6	7
$PERC$	26.54	21.42	16.38	8.57	3.24	1.24	0.56	0.29



(a) (b)

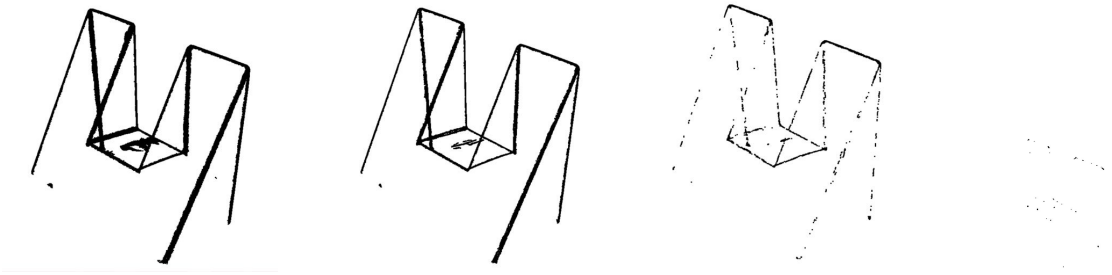
Obrázek 4.23: Model 1: (a) graf dat tabulky 4.4, (b) logaritmická regrese dat.



(a) (b) (c)

Obrázek 4.24: Model 1: (a) vstup, (b) $\theta_{ST} = 3.28$, (c) 3.24% pro $\theta_{ST} = 4$.

Nalezené body modelu 2

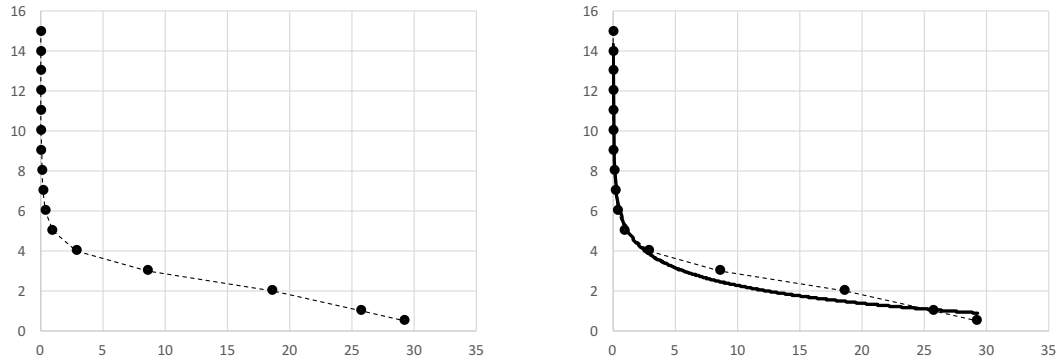


(a) (b) (c) (d)

Obrázek 4.25: Model 2: (a) $\theta_{ST} = 0.5$, (b) $\theta_{ST} = 2$, (c) $\theta_{ST} = 4$, (d) $\theta_{ST} = 7$.

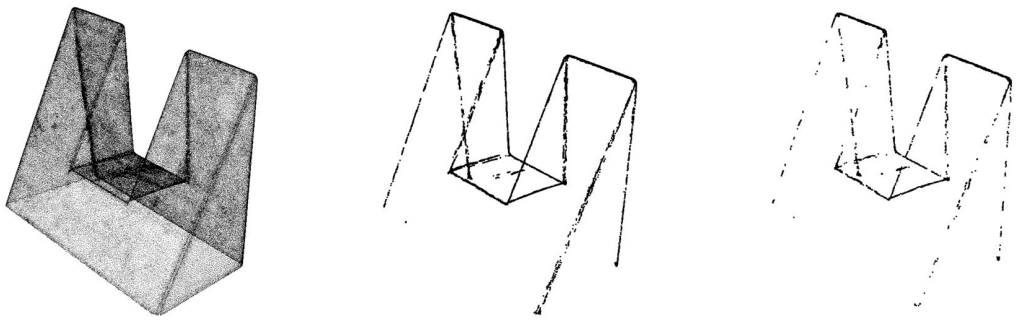
Tabulka 4.5: Tabuka θ_{ST} a $PERC$ pro mračno modelu 2.

θ_{ST}	0.5	1	2	3	4	5	6	7
$PERC$	29.27	25.80	18.65	8.60	2.87	0.95	0.38	0.21



(a) (b)

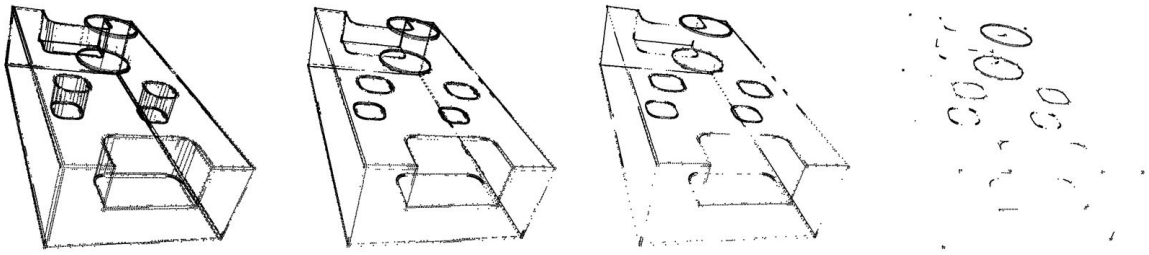
Obrázek 4.26: Model 2: (a) graf dat tabulky 4.5, (b) logaritmická regrese dat.



(a) (b) (c)

Obrázek 4.27: Model 2: (a) vstup, (b) $\theta_{ST} = 3.28$, (c) 2.87% pro $\theta_{ST} = 4$.

Nalezené body modelu kompozitu

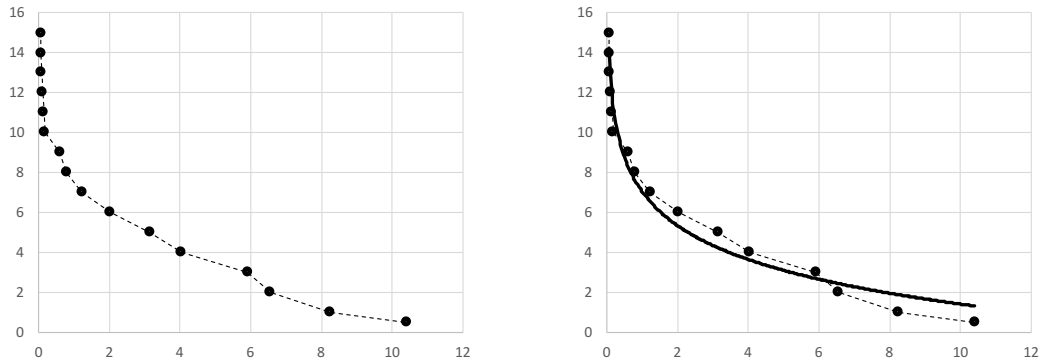


(a) (b) (c) (d)

Obrázek 4.28: Kompozit: (a) $\theta_{ST} = 0.5$, (b) $\theta_{ST} = 2$, (c) $\theta_{ST} = 4$, (d) $\theta_{ST} = 7$.

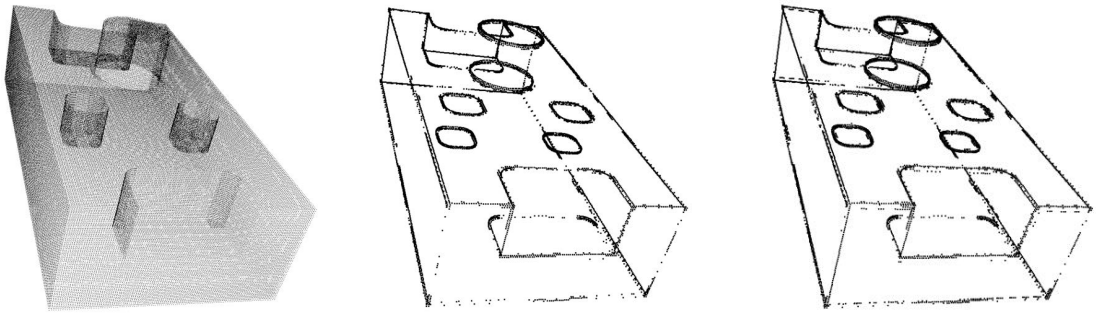
Tabulka 4.6: Tabuka θ_{ST} a $PERC$ pro mračno kompozit.

θ_{ST}	0.5	1	2	3	4	5	6	7
$PERC$	10.41	8.26	6.56	5.91	4.04	3.15	2.03	1.23



(a) (b)

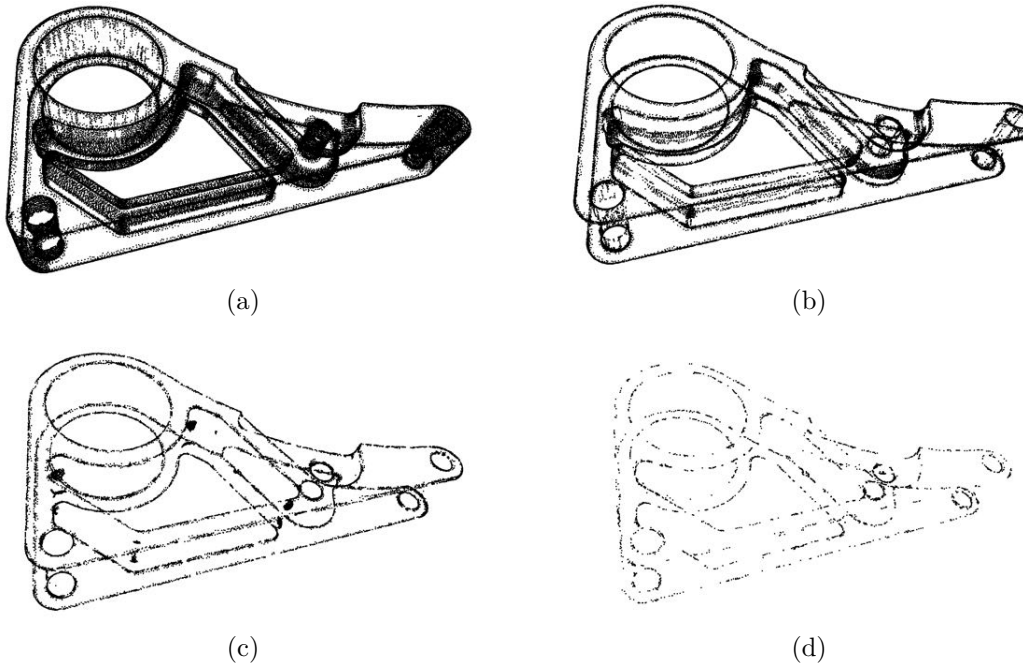
Obrázek 4.29: Kompozit: (a) graf dat tabulky 4.6, (b) logaritmická regrese dat.



(a) (b) (c)

Obrázek 4.30: Kompozit: (a) vstup, (b) $\theta_{ST} = 3.28$, (c) 5.91% pro $\theta_{ST} = 3$.

Nalezené body modelu konzole



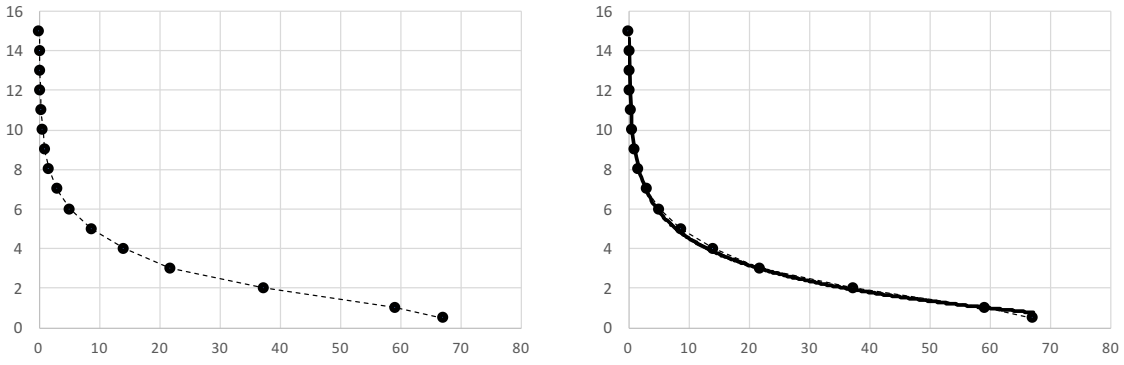
(a) (b)

(c) (d)

Obrázek 4.31: Konzole: (a) $\theta_{ST} = 0.5$, (b) $\theta_{ST} = 2$, (c) $\theta_{ST} = 4$, (d) $\theta_{ST} = 7$.

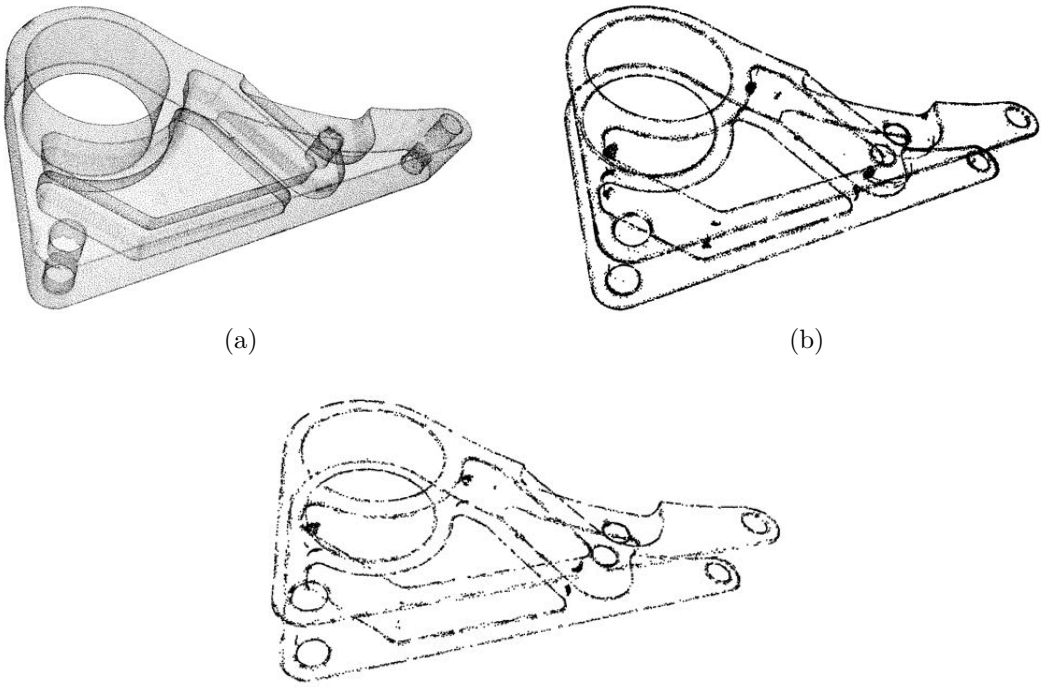
Tabulka 4.7: Tabuka θ_{ST} a $PERC$ pro mračno konzoli.

θ_{ST}	0.5	1	2	3	4	5	6	7
$PERC$	67.14	59.06	37.27	21.70	14.10	8.72	5.09	2.92



(a) (b)

Obrázek 4.32: Konzole: (a) graf dat tabulky 4.7, (b) logaritmická regrese dat.

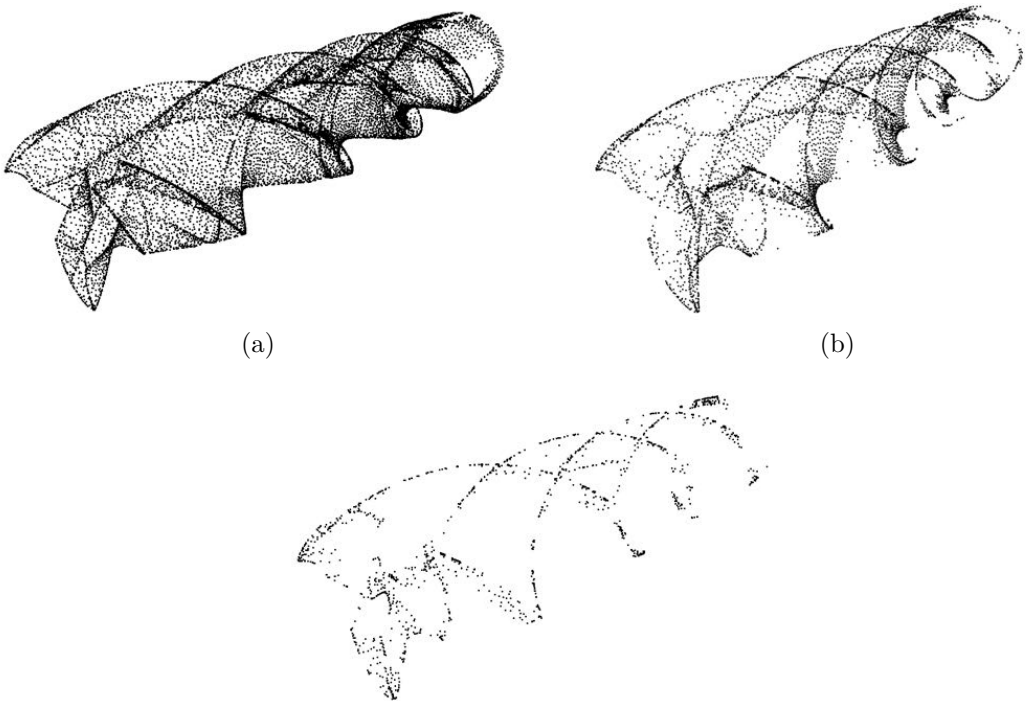


(a) (b)

(c)

Obrázek 4.33: Konzole: (a) vstup, (b) $\theta_{ST} = 3.28$, (c) 8.72% pro $\theta_{ST} = 5$.

Nalezené body modelu frézy



(a) (b)

(c)

Obrázek 4.34: Fréza: (a) vstup, (b) $\theta_{ST} = 3.28$, (c) 5.27% pro $\theta_{ST} = 6$.

Porovnání výsledků detekce u jednotlivých modelů a odhad průměrné regresní funkce

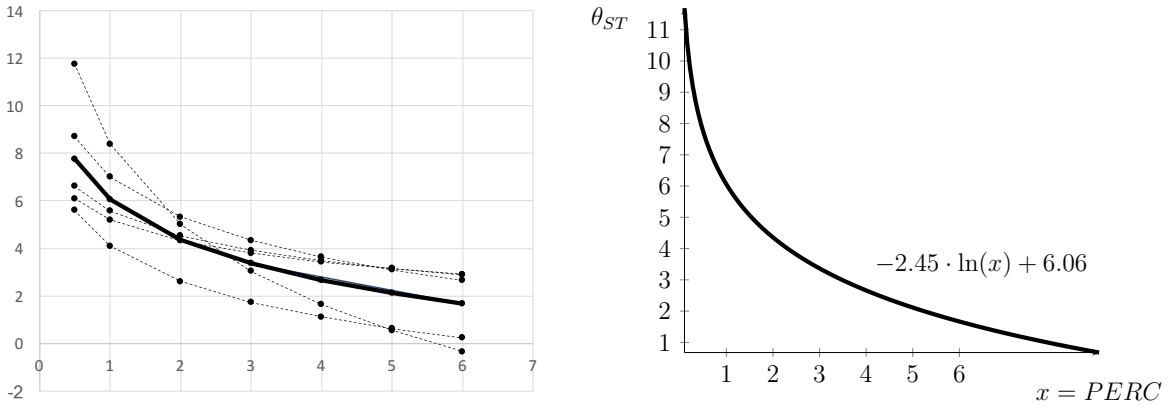
Z výsledků testů a příslušných grafů závislosti θ_{ST} na $PERC$ byla vypozorována logaritmická závislost, která je u jednotlivých výsledků určena a zobrazena. Regresní funkce pro jednotlivé modely, kde x představuje proměnnou $PERC$ jsou ve tvaru:

- **krychle**: $-2.16 \cdot \ln(x) + 4.10$
- **válec**: $-4.87 \cdot \ln(x) + 8.39$
- **model 1**: $-1.50 \cdot \ln(x) + 5.57$
- **model 2**: $-1.28 \cdot \ln(x) + 5.21$
- **kompozit**: $-2.43 \cdot \ln(x) + 7.01$
- **konzole**: $-1.92 \cdot \ln(x) + 9.07$

Jednotlivé regresní funkce jsem porovnal v rozmezí $0.5 \leq x \leq 6$ a vytvořil průměrnou regresní funkci $R(x)$ (obr. 4.35(b)), která má předpis

$$\theta_{ST} \approx R(x) = -2.45 \cdot \ln(x) + 6.06, \tag{4.11}$$

kde hodnota -2.45 je určena jako aritmetický průměr z koeficientů před $\ln(x)$, tedy průměr z $\{-4.87, -2.43, -2.16, -1.92, -1.50, -1.28\}$. Hodnota posunu logaritmické funkce v ose y, 6.06 je určena jako aritmetický průměr z čísel $\{4.10, 5.21, 5.57, 7.01, 8.39, 9.07\}$.



(a) (b)

Obrázek 4.35: (a) porovnání grafů regresních funkcí pro jednotlivé modely s výsledným průměrným regresním modelem, (b) graf průměrné logaritmické regresní funkce.

Ze získané funkce $R(x)$ lze nyní odvodit hodnotu parametru θ_{ST} pro výslednou segmentaci s 0,5–6% nalezenými body. Tedy

$$\begin{aligned} R(0.5) = -2.45 \cdot \ln(0.5) + 6.06 &\leq \theta_{ST} \leq -2.45 \cdot \ln(6) + 6.06 = R(6) \\ 7.75 &\leq \theta_{ST} \leq 1.67. \end{aligned} \tag{4.12}$$

Výchozí hodnotu θ_{ST} pro můj navržený algoritmus jsem určil jako hodnotu regresní funkce $R(x)$ v $x = 3.1$ – průměrná hodnota $\{0.5, 1, 2, 3, 4, 5, 6\}$ a získal tím počáteční hodnotu $\theta_{ST} \doteq 3.28$. Tato hodnota byla také použita při segmentování testovacích mračen.

Navržený algoritmus tedy ve výchozím nastavení $\theta_{ST} = 3.28$ nalezne ve vstupním mračnu body ležící na plochách s lokální změnou normál do 3.28°, které odstraní. Jestliže je nalezený počet bodů v procentech menší než požadovaná hodnota 3–6%, je třeba hodnotu θ_{ST} snížit. V opačném případě se hodnota θ_{ST} zvýší. Tedy algoritmus lze provést také iteračně. V každém kroku se určí hodnota $PERC$ a porovná s předurčenou prahovou hodnotou θ_{PERC}. Pokud $PERC \approx \theta_{PERC}$ – algoritmus vrátí aktuální výsledek; jestliže $PERC < \theta_{PERC}$ – zmenší se θ_{ST}; a v případě $PERC > \theta_{PERC}$ se θ_{ST} zvětší. Proces se opakuje, dokud není dosaženo požadované procentuální zastoupení nalezených bodů specifického rysu. Tento postup lze zapsat následujícím pseudokódem v Algoritmu 8.

Algoritmus 8 Feature detection

```
1: procedure FeatureDetection(thetaST, thetaCT, thetaPST, thetaPERC, cloudIN, cloudOUT) ▷ Algoritmus detekuje body specifického rysu ve vstupním mračnu bodů
2:     načtení souboru cloudIN
3:     NOPI(cloudIN)                                  ▷ počet bodů vstupního mračna bodů
4:     RegionGrowing(thetaPST, thetaCT, cloudIN, cloudOUT)   ▷ počáteční segmentace
5:     NOPO(cloudOUT)                                 ▷ počet nalezených bodů
6:     PERC = (NOPO / NOPI) * 100                     ▷ procentuální zastoupení nalezených bodů
7:     thetaPERC = 3.1                                ▷ práh procentuálního zastoupení
8:     if (PERC = thetaPERC) then
9:         return cloudOUT
10:    else
11:        if (PERC < thetaPERC) then
12:            while (PERC < thetaPERC) do
13:                thetaST = thetaST – 0.5
14:                cloudIN = cloudOUT
15:                NOPI(cloudIN
16:                RegionGrowing(thetaST, thetaCT, cloudIN, cloudOUT)
17:                NOPO(cloudOUT)
18:                PERC = (NOPO / NOPI) * 100
19:            end while
20:        else
21:            if (PERC > thetaPERC) then
22:                while (PERC > thetaPERC) do
23:                    thetaST = thetaST + 0.5
24:                    cloudIN = cloudOUT
25:                    NOPI(cloudIN)
26:                    RegionGrowing(thetaST, thetaCT, cloudIN, cloudOUT)
27:                    NOPO(cloudOUT)
28:                    PERC = (NOPO / NOPI) * 100
29:                end while
30:            end if
31:        end if
32:    end if
33:    PCLVisualizer(cloudOUT)                        ▷ Zobrazení nalezených bodů
34:    return cloudOUT
35: end procedure
```

4.4. Proklad křivkou

Tato část textu se věnuje prokladu nalezených bodů specifického rysu b-spline křivkou. Volba tohoto typu reprezentace křivky je v oblasti počítačové grafiky běžná a ve výpočetní technice je relativně snadno zpracovatelná. Knihovna PCL obsahuje skupinu funkcí s označením **FittingCurve**. Ve svém algoritmu používám upravenou verzi funkce **initNurbsCurvePCA**, což je implementace metody popsané Thomasem Mörwaldem *et al.* v roce 2016 [41] která je upravenou verzí přístupu publikovaného v roce 2006 Wangem *et al.* [64].

4.4.1. Princip metody prokladu

Algoritmus prokladu vytváří aproximovanou křivku pro zadané body iterativně. Vstupními parametry jsou: požadovaný stupeň křivky, počet řídicích bodů a interně také maximální chyba aproximace. V prvním kroku se vytvoří okrajová křivka – podobná kružnici. Spočte se chyba aproximace pro každý zadaný bod a v místě, kde je chyba mimo určené maximum, se přidá další uzel a dopočítá se nový řídicí bod. Postup přidávání řídicích bodů se opakuje, dokud není chyba v zadaném rozmezí nebo není dosaženo požadovaného počtu řídicích bodů (podle toho, co nastane dříve).

Tvar hledané racionální b-spline křivky

Mějme dány vstupní body $Q_k \in \mathbb{R}^3, k = 0, 1, 2, \ldots, m$, stupeň hledané křivky $p \geq 1$ a určený počet řídicích bodů n $(m > n, n \geq p)$. Hledáme b-spline křivku ve tvaru

$$C(u) = \sum_{i=0}^{n} N_{i,p}(u) P_i, \text{ kde } u \in \langle 0, 1 \rangle, \tag{4.13}$$

kde $N_{i,p}(u)$ jsou b-spline bázové funkce, P_i hledané řídicí body. Pro nalezení řídicích bodů P_i je nutné určení parametrů $\{\bar{u}_k\} \in \langle 0, 1 \rangle, k = 0, 1, 2, \ldots, m$ takových, že $Q_k = C(\bar{u}_k)$. A také uzlový vektor $U = \{u_i\}, i = 0, 1, 2, \ldots, n + p + 1$.

Určení parametrů $\bar{u}_k$

Parametry $\{\bar{u}_k\}$ určím metodou ekvidistantního rozdělení (viz kap. 3.4.3 str. 48), tedy

$$\begin{aligned} &\bar{u}_0 = 0 \quad \bar{u}_n = 1 \\ &\bar{u}_k = \tfrac{k}{n} \quad \text{pro } k = 1, 2, \ldots, n-1. \end{aligned} \tag{4.14}$$

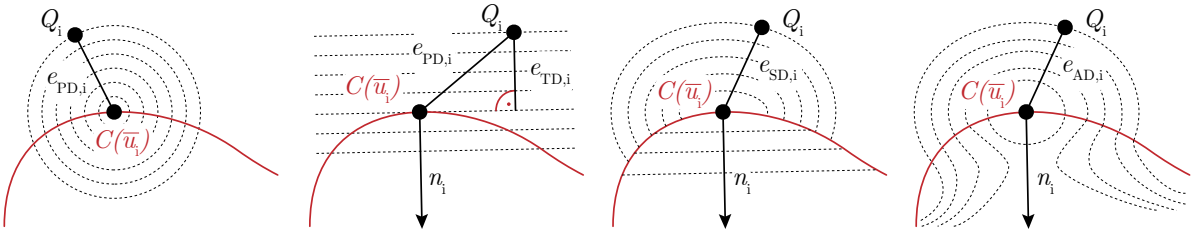
Určení uzlového vektoru U

Uzlový vektor $U = \{u_0, u_1, \ldots, u_{n+p+1}\}$ určím metodou rovnoměrného rozdělení (viz kap. 3.4.3 str. 48)

$$\begin{aligned} &u_0 = \cdots = u_p = 0 \quad u_{m-p} = \cdots = u_m = 1 \\ &u_{j+p} = \tfrac{j}{n-p+1} \qquad \text{kde } j = 1, 2, \ldots, n-p. \end{aligned} \tag{4.15}$$

Určení chyby aproximace v bodě Q_k

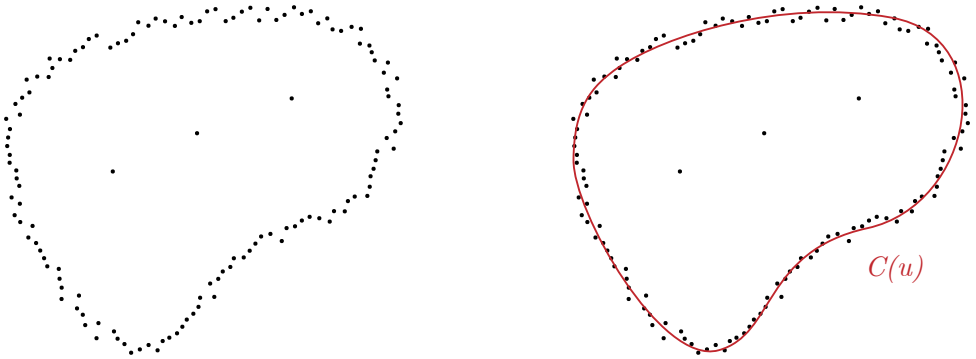
Chybu aproximace e_i mezi bodem Q_i a $C(\bar{u}_i)$ určím jako váženou vzdálenost typu *squared distance – SD* – $e_{SD,i}$ (viz kap. 3.4.5 str. 55), která je v této metodě označena jako *asymetrická vzdálenost – AD*. Asymetrická vzdálenost kombinuje výhody principu měření vzdálenosti metodou bodové vzdálenost, tečné vzdálenosti a squared distance. Porovnání těchto metod měření vzdálenosti bodu od prokládané křivky je znázorněna na obr. 4.36.



(a) (b) (c) (d)

Obrázek 4.36: Ilustrace vzdálenosti e_i (a) PD s izočarami pro $e_{PD,i}$, (b) TD s izočarami pro $e_{TD,i}$, (c) SD s izočarami pro $e_{SD,i}$, (d) AD s izočarami pro $e_{AD,i}$.

Hlavní myšlenkou této metriky je přiřazení větší váhy bodům na okraji vstupní množiny než bodů, které leží při vnitřní části – ty mohou být chybové, reziduální z předchozích zpracování či způsobené šumem. Proto tedy chci, aby se aproximovaná křivka iterativně přibližovala více k bodům na okraji (viz obr. 4.37). Tento přístup je pro moji metodu detekce ideální – mám totiž body na okraji původních stěn a jimi chci proložit křivku; pokud tedy zůstanou reziduální body uvnitř původních ploch, budou tímto principem ignorovány.



(a) (b)

Obrázek 4.37: (a) vstupní body, (b) proložená křivka.

Asymetrická chyba $e_{AD,i}$ mezi původním bodem Q_i a vypočítaným bodem křivky $C(\bar{u}_i)$ je definována následovně.
Nechť

$$e_{PD,i} := \sqrt{(Q_i - C(\bar{u}_i))^2} = \sqrt{d_i^2} = ||d_i||, i = 0, 1, 2, \ldots, m, \tag{4.16}$$

dále nechť

$$e_{TD,i} := d_i^T n_i, i = 0, 1, 2, \ldots, m, \tag{4.17}$$

kde n_i je jednotkový normálový vektor v $C(\bar{u}_i)$ a nechť

$$e_i^2 = e_{SD,i}^2 := \begin{cases} \frac{e_{TD,i}}{e_{TD,i}-\rho_i}(d_i^T t_i)^2 + (d_i^T n_i)^2 & \text{pro } e_{TD,i} < 0 \\ \\ e_{TD,i}^2 & \text{pro } 0 \leq e_{TD,i} < \rho_i, \end{cases} \quad i = 0, 1, 2 \ldots, m, \tag{4.18}$$

kde ρ_i je poloměr křivosti v $C(\bar{u}_i)$, tedy $\rho_i = ||C''(\bar{u}_i)||$ a t_i je jednotkový tečný vektor v bodě $C(\bar{u}_i)$, pak

$$e_{AD_i}^2 = w_a(d_i)e_i^2, \qquad \text{kde } w_a(d_i) := \begin{cases} e^{-\frac{d_i^2}{\sigma^2}} & \text{pro } d_i < 0 \\ \\ 1 & \text{pro } d_i \geq 0 \end{cases} \quad i = 0, 1, 2 \ldots, m \tag{4.19}$$

je vážená asymetrická chyba aproximace. Kde σ představuje velikost posunu váhové funkce vzhledem k orientované vzdálenosti (obr. 4.38(a)). Tedy bodům se záporně orientovanou vzdáleností $d_i = e_{TD,i}$ přiřadíme zmenšenou chybu aproximace a bodům s kladnou vzdáleností přiřadíme nezměněnou chybu. Pro zlepšení výsledků algoritmu jsem navrhl změnu a namísto Eulerova čísla e využívám hodnotu zlatého řezu $\varphi \doteq 1.6180339$. Tedy v mojí upravené verzi algoritmu používám váženou chybu $e_{\varphi,i}^2$ ve tvaru

$$e_{\varphi,i}^2 = w_{\varphi,i}(d_i)e_i^2, \qquad \text{kde } w_\varphi(d_i) := \begin{cases} \left(\frac{1}{\varphi} + \varphi\right)^{-\frac{d_i^2}{\sigma^2}} & \text{pro } d_i < 0 \\ \\ 1 & \text{pro } d_i \geq 0 \end{cases} \quad i = 0, 1, 2 \ldots, m, \tag{4.20}$$

přičemž základy mocnin obou vah jsou v tomto vztahu:

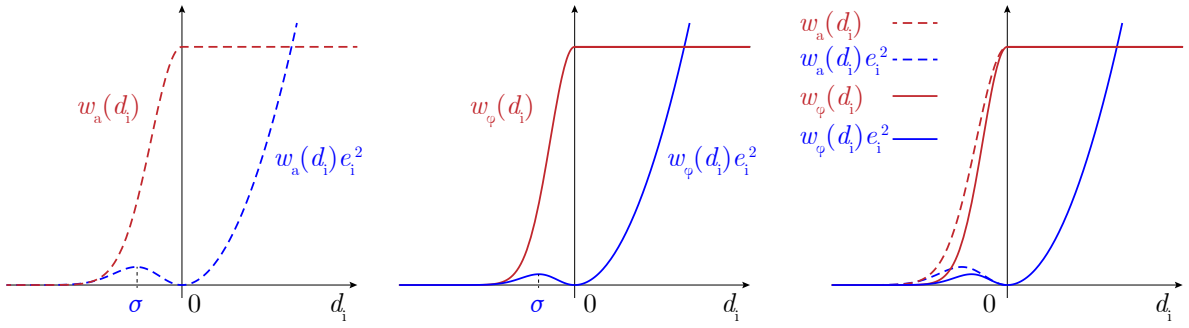
$$\left(\frac{1}{\varphi} + \varphi\right) \doteq 2.2360679 < e. \tag{4.21}$$

Při využití této upravené váhy se křivka ještě více přibližuje k okrajovým bodům. Porovnání vah $w_a(d_i), \bar{w}_a(d_i)$ a váhových funkcí $w_a(d_i)e_i^2, \bar{w}_a(d_i)e_i^2$ pro body ležící na přímce ve směru normály křivky C je zobrazeno na obr. 4.38.

Minimalizace cílové funkce

Cílová funkce f je tedy tvaru

$$f(P_j) = \sum_{i=0}^{m} e_{\varphi,i}^2 + f_s(P_j) + f_c(P_j), \tag{4.22}$$



(a) (b) (c)

Obrázek 4.38: (a) původní váha a vážená vzdálenost, (b) nová navržená váha a vážená vzdálenost, (c) porovnání.

kde $f_s(P_j)$ je vyhlazovací funkce tvaru

$$f_s(P_j) = w_s \int ||C''(\bar{u}_i)||^2 = w_s \int \rho^2, \tag{4.23}$$

přičemž $w_s \in \mathbb{R}^+$ je váha vyhlazovací funkce a $f_c(P_j)$ je funkce zajišťující správné chování křivky v místech s vysokou křivostí – zaručuje vložení nového uzlu v místě s vysokou chybou aproximace (více v [41]).

Chyba aproximace $e^2_{AD,i}$ se v této metodě určuje kromě míst $C(\bar{u}_i)$ určených parametry $\{\bar{u}_k\}$ také v místech $C(\epsilon_k)$, které se určí jako poloviny mezi jednotlivými uzly, tedy

$$\epsilon_k = \frac{u_k + u_{k+1}}{2}, \text{pro } k = 1, 2, 3, \ldots, n+p, \tag{4.24}$$

kdy se chyba aproximace v tomto místě určuje jako vzdálenost $C(\epsilon_k)$ od nejbližšího bodu Q_{ϵ_k} z množiny vstupních bodů. Tedy chybu v tomto místě určíme takto

$$||Q_{\epsilon_k} - C(\epsilon_k)||, k = 1, 2, 3, \ldots, n+p \tag{4.25}$$

a funkce $f_c(P_j)$ má tedy tvar

$$f_c(P_j) = w_c \sum_{k=1}^{n+p} ||Q_{\epsilon_k} - C(\epsilon_k)||^2 \tag{4.26}$$

kde $w_c \in \mathbb{R}^+$ je její určená váha. Cílová funkce má tento úplný tvar

$$f(P_j) = \sum_{i=0}^{m} e^2_{\varphi,i} + w_s \int \rho^2 + w_c \sum_{k=1}^{n+p} ||Q_{\epsilon_k} - C(\epsilon_k)||^2. \tag{4.27}$$

Minimalizací této cílové funkce vzhledem k neznámým P_j určíme řídicí body hledané křivky. Následně spočítáme chybu aproximace v polovinách mezi uzlovými body – v místech ϵ_k. Pokud je chyba v místě $\epsilon_k > E$, vloží se v tomto místě nový uzel, upraví se cílová funkce a dopočítají nové potřebné řídicí body. Postup se opakuje, dokud není chyba ve všech místech v určené mezi E. Princip lze popsat Algoritmem 9.

Navržená metoda vstupní body aproximuje uzavřenou b-spline křivkou. Data z předchozí segmentace je tedy potřeba rozdělit na jednotlivé části, ve kterých se následně provede proklad křivkou. Data lze rozdělit buď pomocí postupné region growing segmentace,

Algoritmus 9 Curve fitting

```
1: procedure CurveFitting(Qi, p, n) ▷ Algoritmus proloží body Qi křivku C stupně
   p s n řídicími body
2:     určení počáteční křivky
3:     while spuštěno do
4:         for pro každý bod Qi do
5:             urči eAD,i
6:         end for
7:         for pro všechny poloviny epsilonK do
8:             najdi nejbližší bod QepsilonK
9:         end for
10:        minimalizace f(Pj)
11:        for pro všechny poloviny epsilonK do
12:            najdi nejbližší bod QepsilonK
13:            if (|| QepsilonK – C(epsilonK) || 2 > E) then
14:                vlož uzel v místě epsilonK
15:            end if
16:        end for
17:    end while
18: end procedure
```

popsané v kap. 4.3.1 (str. 68), nebo pomocí softwaru třetích stran pro manuální rozdělení mračna bodů na jednotlivé plochy. V případě segmentace v daném kroku rozdělím jednotlivé body specifických rysů do příslušných částí vstupního mračna bodů. Tento postup je proveditelný, ale vyžaduje v každém kroku manuální odhad správného parametru θ_{ST} tak, aby byly označeny pouze body z dané stěny – plochy. Proto pro ilustraci metody prokladu použiji stěnu krychle, válce, dále mračno osmicípé rovinné hvězdy a dále vybrané testovací mračna modelu 1, které jsem ručně vybral z testovacích mračen.

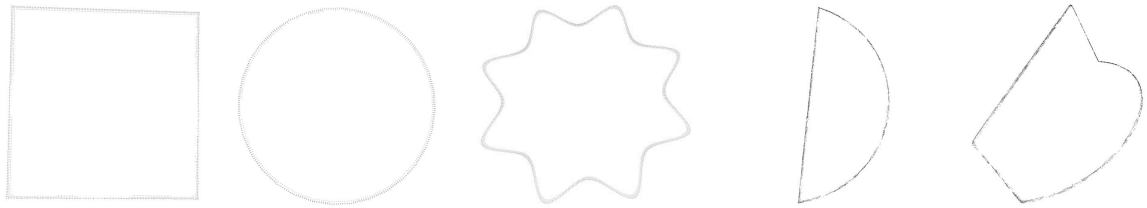
4.4.2. Testy algoritmu prokladu b-spline křivkou

V této části jsou v prostředí **PCLVisualizer** znázorněny výsledky algoritmu prokladu b-spline křivkou.

Vstupní data

Jako vstupní data algoritmu prokladu křivkou jsem použil uzavřené celky nalezených bodů specifických rysů z předchozí části a také uměle vytvořená mračna v programu Blender. Jmenovitě: podstavu krychle a válce, mračno rovinné osmicípé hvězdy a části hran modelu 1 (viz obr. 4.39). Podstava krychle – čtverec o straně $200mm$, podstava válce – kružnice s průměrem $200mm$, osmicípá hvězda s poloměrem opsané kružnice $200mm$, hrana 1 o rozměrech opsaného obdélníka $200 \times 80mm$ a hrana 2 o rozměrech opsaného kvádru (délka, výška, šířka) $200 \times 80 \times 80mm$. Proklad jsem následně provedl pro různé počty řídicích bodů hledané b-spline křivky.

V následující části jsou zobrazeny výsledky prokladu pro jednotlivá testovací mračna bodů. Vždy je uvedeno několik výsledků pro různé množství řídicích bodů s tabulkou prů-



(a) (b) (c) (d) (e)

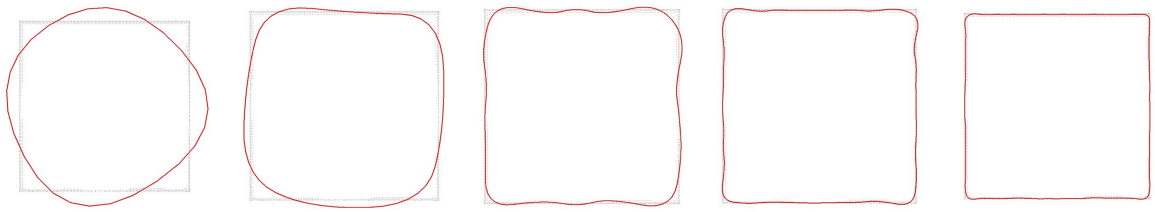
Obrázek 4.39: Testovací mračna bodů: (a) čtverec, (b) kružnice, (c) hvězda, (d) hrana 1, (e) hrana 2.

Tabulka 4.8: Vstupní mračna s příslušným počtem bodů NOP (*number of points*).

model	čtverec	kružnice	hvězda	hrana 1	hrana 2
NOP	1 632	1 307	2 560	2 477	3 130

měrné eukleidovské chyby aproximace e_ϕ. Optimální aproximaci jsem zvolil při průměrné chybě $e_\phi \in \langle 0, 1 \rangle\ mm$. Následně je zobrazeno srovnání vstupu a výstupu.

Výsledky prokladu mračna čtverce



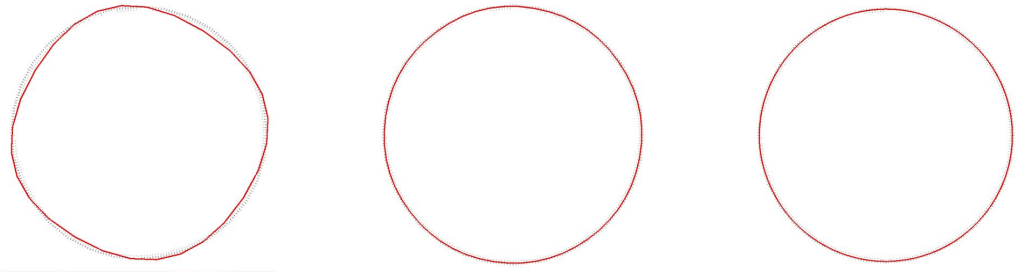
(a) (b) (c) (d) (e)

Obrázek 4.40: Proklad pro různý počet řídicích bodů: (a) 4, (b) 10, (c) 20, (d) 40, (e) 80.

Tabulka 4.9: Průměrná chyba aproximace e_ϕ pro mračno čtverce.

počet řídicích bodů	4	10	20	40	80
průměrná chyba e_ϕ [mm]	14.01	5.54	2.02	1.25	0.99

Výsledky prokladu mračna kružnice



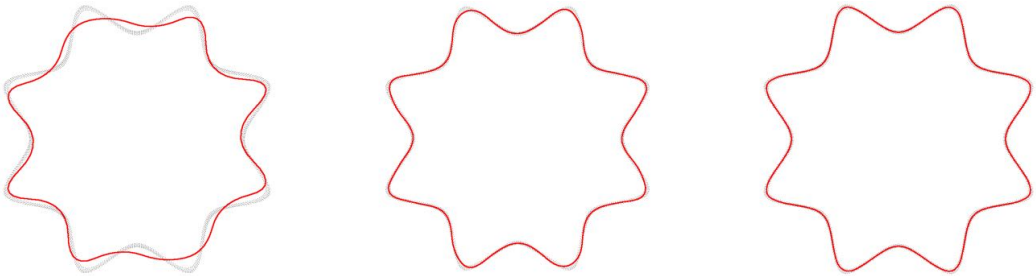
(a) (b) (c)

Obrázek 4.41: Proklad pro různý počet řídicích bodů: (a) 4, (b) 10, (c) 20.

Tabulka 4.10: Průměrná chyba aproximace e_ϕ pro mračno kružnice.

počet řídicích bodů	4	10	20
průměrná chyba e_ϕ [mm]	2.35	1.16	0.83

Výsledky prokladu mračna osmicípé hvězdy



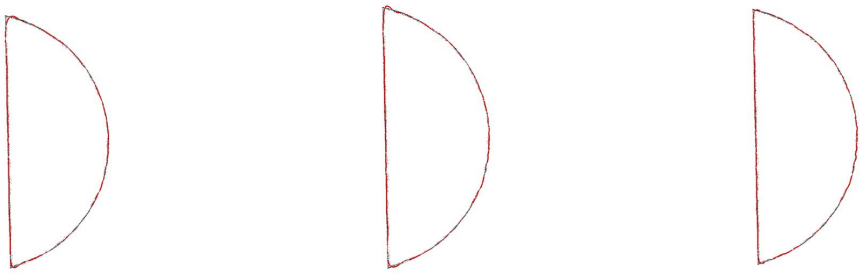
(a) (b) (c)

Obrázek 4.42: Proklad pro různý počet řídicích bodů: (a) 20, (b) 40, (c) 80.

Tabulka 4.11: Průměrná chyba aproximace e_ϕ pro mračno osmicípé hvězdy.

počet řídicích bodů	20	40	80
průměrná chyba e_ϕ [mm]	3.96	0.96	0.85

Výsledky prokladu mračna hrany 1



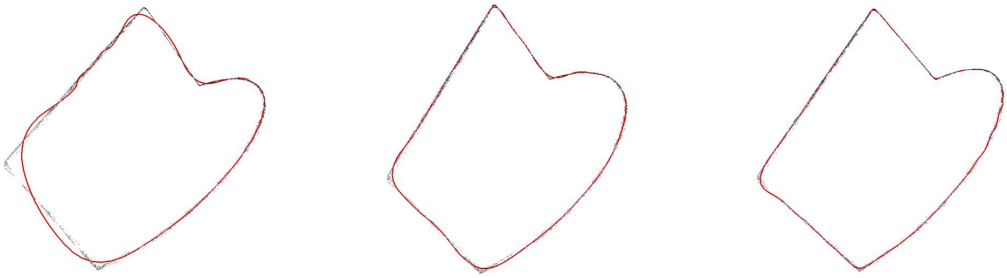
(a) (b) (c)

Obrázek 4.43: Proklad pro různý počet řídicích bodů: (a) 80, (b) 100, (c) 160.

Tabulka 4.12: Průměrná chyba aproximace e_ϕ pro mračno hrany 1.

počet řídicích bodů	80	100	160
průměrná chyba e_ϕ [mm]	0.41	0.37	0.27

Výsledky prokladu mračna hrany 2



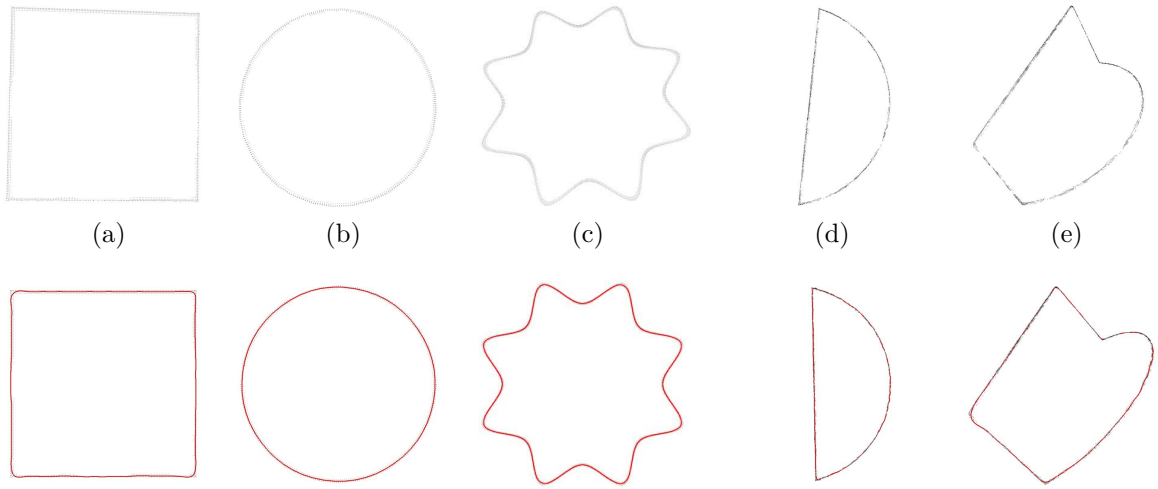
(a) (b) (c)

Obrázek 4.44: Proklad pro různý počet řídicích bodů: (a) 10, (b) 40, (c) 80.

Tabulka 4.13: Průměrná chyba aproximace e_ϕ pro mračno hrany 2.

počet řídicích bodů	10	40	80
průměrná chyba e_ϕ [mm]	2.19	0.67	0.33

Porovnání výsledků prokladu s původním mračnem



Obrázek 4.45: Porovnání: (a) – (e) vstup, (f) – (j) výstup prokladu.

4.5. Porovnání s jinými metodami

Následující část práce porovnává navržené algoritmy s metodami vybraných autorů, kteří se věnují stejné problematice. Je rozdělena do dvou částí dle řešené problematiky: detekce bodů ostrých hran; proklad křivkou.

4.5.1. Detekce bodů na specifických rysech

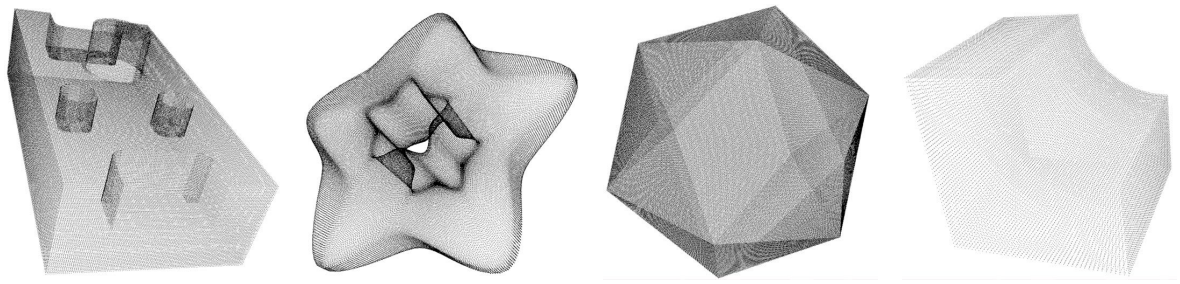
V této části jde o porovnání s metodou, kterou představil Demaris *et al.* [15] z roku 2007; a dále s metodou navrženou Weberem *et al.* [65] z roku 2010.

Demaris *et al.* [15] v prvním kroku segmentuje vstupní mračno pro odstranění pouze rovinných bodů, a tím získá kandidáty bodů ostrých hran, které následně zpracovává jako graf pro extrakci hledaných bodů. Porovnání metod je znázorněno pro model kompozitu.

Weber *et al.* [65] používá jako první krok také segmentaci pro odstranění rovinných bodů. V druhém kroku využívá Gaussovo zobrazení pro extrakci a klasifikaci hledaných bodů ostré hrany. Porovnání je provedeno prostřednictvím modelů hvězdy, dvacetistěnu a seříznuté krychle.

Testovací mračna

Porovnání s ostatními metodami bylo provedeno prostřednictvím modelů kompozitu, hvězdy, dvacetistěnu a seříznuté krychle (obr. 4.46).

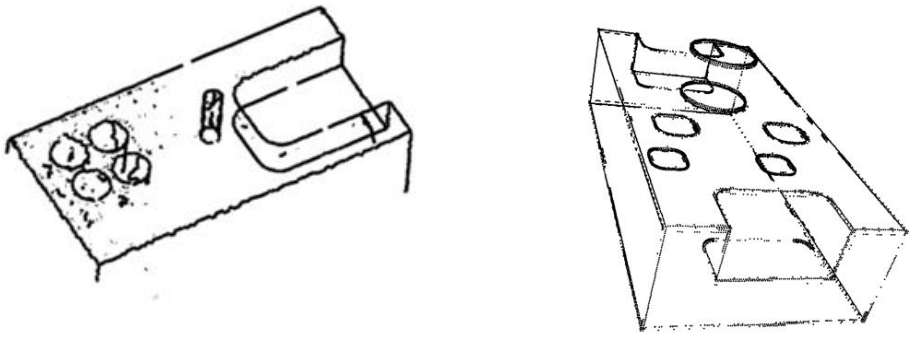


(a) (b) (c) (d)

Obrázek 4.46: Modely pro porovnání (a) kompozit, (b) hvězda, (c) dvacetistěn, (d) seříznutá krychle.

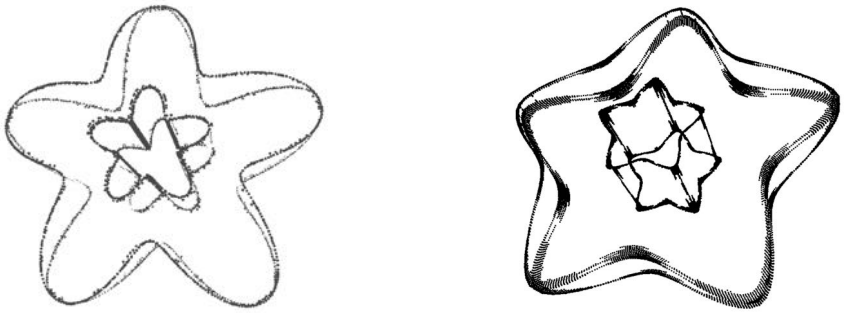
Porovnání

V porovnání s metodou Demaris [15] pro podobná data moje metoda poskytuje přijatelnější řešení (obr. 4.47). Viditelné rozdíly jsou v částech válcovitých výstupků tělesa, kde metoda Demaris vykazuje chybnou detekci. Tento nedostatek se však v porovnávané metodě následně řeší vyhlazováním.



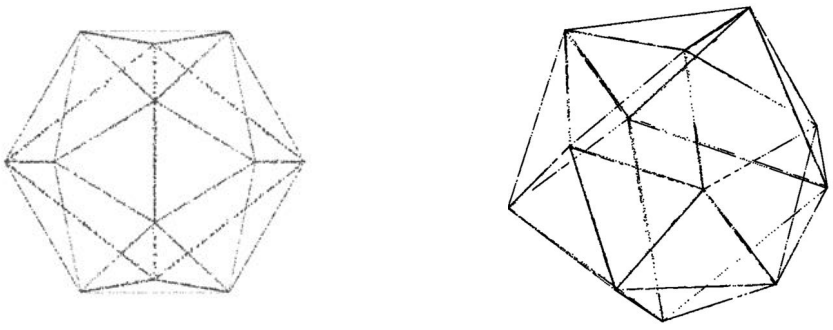
(a) (b)

Obrázek 4.47: Porovnání s metodou Demaris [15] (a) Demaris, (b) vlastní navržená metoda pro $\theta_{ST} = 4$.



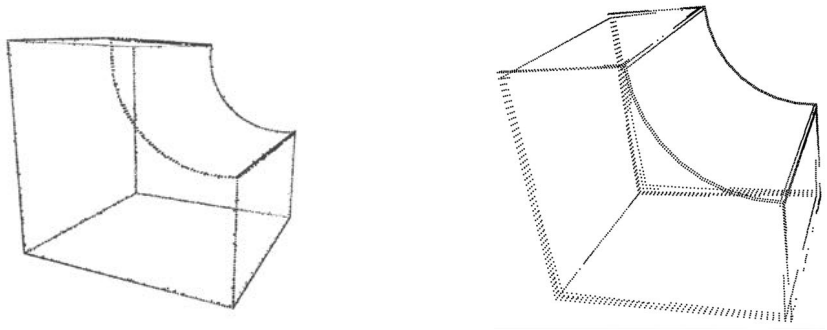
(a) (b)

Obrázek 4.48: Porovnání s metodou Weber [65] (a) Weber, (b) vlastní navržená metoda pro $\theta_{ST} = 1.2$.



(a) (b)

Obrázek 4.49: Porovnání s metodou Weber [65] (a) Weber, (b) vlastní navržená metoda pro $\theta_{ST} = 1$.



(a) (b)

Obrázek 4.50: Porovnání s metodou Weber [65] (a) Weber, (b) vlastní navržená metoda pro $\theta_{ST} = 1.5$.

V porovnání s metodou Weber [65] moje metoda poskytuje srovnatelné výsledky pro všechny testovací mračna (obr. 4.48, 4.49, 4.50). V některých místech však moje navržená metoda vykazuje menší množství detekovaných bodů ostrých hran. Důvodem tohoto malého množství nalezených bodů je umělé vytvoření modelů v programu Blender, který mračna bodů pouze simuluje.

4.5.2. Proklad křivkou

V této části textu je provedeno porovnání algoritmu Mörwald [41] z roku 2016 s mojí navrženou upravenou verzí v kap. 4.4.1 (str. 83). Obě metody se liší ve způsobu, kterým určují váhu vzdáleností jednotlivých bodů od hledané křivky. Tudíž i cílová funkce pro minimalizaci metodou nejmenších čtverců je odlišná. V původní metodě je použita tato cílová funkce s příslušnou váhou $w_a(d)$ v tomto tvaru

$$f(P_j) = \sum_{i=0}^{m} e_{AD,i}^2 + f_s(P_j) + f_c(P_j), \tag{4.28}$$

kde

$$e_{AD_i}^2 = w_a(d_i)e_i^2, \qquad \text{kde } w_a(d_i) := \begin{cases} e^{-\frac{d_i^2}{\sigma^2}} & \text{pro } d_i < 0 \\ 1 & \text{pro } d_i \geq 0 \end{cases} \quad i = 0, 1, 2 \ldots, m. \tag{4.29}$$

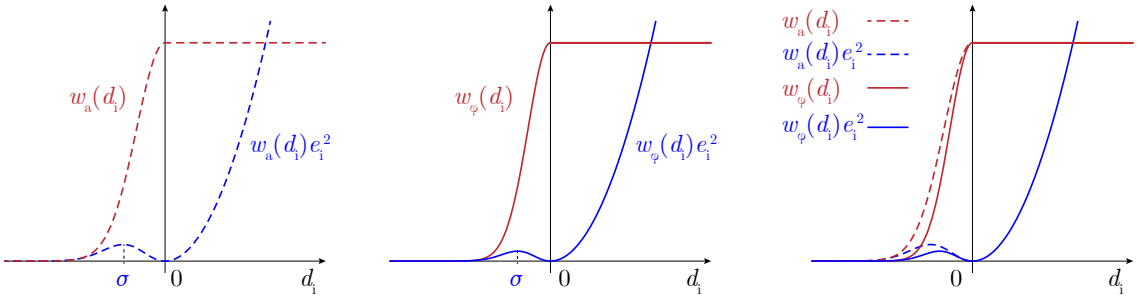
Mnou navrhovaná cílová funkce s příslušnou váhou $w_\varphi(d)$ má tvar

$$f(P_j) = \sum_{i=0}^{m} e_{\varphi,i}^2 + f_s(P_j) + f_c(P_j), \tag{4.30}$$

kde

$$e^2_{\varphi,i} = w_{\varphi,i}(d_i)e_i^2, \qquad \text{kde } w_\varphi(d_i) := \begin{cases} \left(\frac{1}{\varphi} + \varphi\right)^{-\frac{d_i^2}{\sigma^2}} & \text{pro } d_i < 0 \\ 1 & \text{pro } d_i \geq 0 \end{cases} \qquad i = 0, 1, 2 \ldots, m, \quad (4.31)$$

Rozdílné váhové funkce jsou porovnány v následujícím obr. 4.51 (viz kap. 4.4, str. 81).

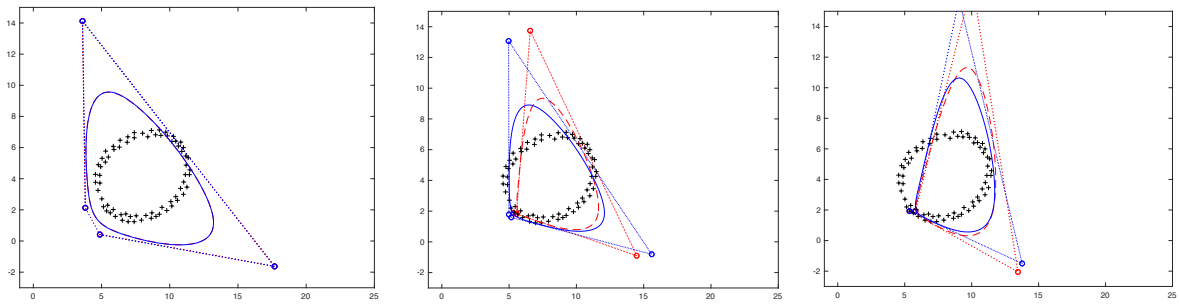


(a) (b) (c)

Obrázek 4.51: (a) původní váha a váhovaná vzdálenost, (b) nová navržená váha a váhovaná vzdálenost, (c) porovnání.

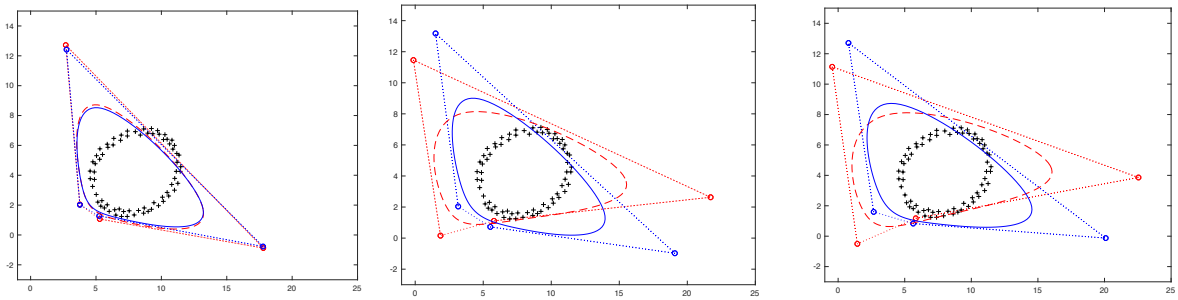
Navržené upravené váhování způsobuje, že se prokládaná křivka více přibližuje k bodům na okraji vstupních bodů. Pro porovnání obou metod jsem nejdříve vstupní body aproximoval počáteční b-spline křivkou pomocí metody popsané v kap. 3.4.3 (str. 48). Následně jsem provedl porovnání první iterace obou algoritmů se shodným počtem řídicích bodů pro různé hodnoty parametru σ (obr. 4.52). Obecně se tento parametr nastavuje s velmi malou hodnotou, při které jsou obě metody srovnatelné. Při porovnání jsem určoval průměrnou chybu aproximace – E_1 pro původní a E_2 pro navrhovanou metodu. V první iteraci, kdy algoritmy pracovaly se stejnou počáteční křivkou, mají obě metody shodné chyby aproximace, ale odlišný tvar křivky. Křivky vzniklé v první iteraci jsem následně použil jako počáteční křivky pro druhou iteraci jednotlivých metod. Porovnal jsem opět výsledky pro různé hodnoty parametru σ včetně chyb E_1, E_2 (obr. 4.53). V tomto kroku vykazovala moje navržená metoda s malou hodnotou σ menší chybu. Porovnání bylo provedeno pomocí matematického programu MATLAB R2017b.

Zřejmě tedy pro malé hodnoty σ jsou obě metody srovnatelné a konvergují k podobné křivce. Moje navržená metoda se k okrajovým bodů zadané množiny přibližuje nepatrně více. V případě navržené metody dosahuji přesnějšího řešení s nižší chybou.



(a) (b) (c)

Obrázek 4.52: Porovnání s metodou Mörwald [41] (červená) v první iteraci (a) $\sigma = 0.02$, (b) $\sigma = 0.5$, (c) $\sigma = 0.9$; ve všech případech $E_1 = E_2 = 0.6489$



(a) (b) (c)

Obrázek 4.53: Porovnání s metodou Mörwald [41] (červená) ve druhé iteraci (a) $\sigma = 0.02$, $E_1 = 0.9387, E_2 = 0.9382$, (b) $\sigma = 0.09$, $E_1 = 0.9661, E_2 = 0.9560$, (c) $\sigma = 0.1$, $E_1 = 0.9702$, $E_2 = 0.9604$.

5. Závěr

Tato disertační práce představuje zcela nový algoritmus určený k automatickému vyhledávání a vizualizaci ostrých přechodů v libovolném mračnu bodů. Algoritmus je založen na postupném odstraňování bodů, které jsou součástí ploch s malou změnou normálových vektorů a křivostí. Body, které po odstranění zůstanou, jsou hledané body ostrých hran. K určení normálových vektorů je využito statistické metody kovarianční matice. Příslušná křivost je vypočítána pomocí vlastních čísel této matice metodou povrchového rozptylu – *surface variation* [47]. Body, v jejichž okolí dochází k malé změně normálových vektorů a křivostí, se vyhledávají a odstraňují pomocí principu *region growing segmentace*. Takové využití region growing segmentace je v navrženém algoritmu jedinečné. Ostatní metody používají segmentaci pro odstranění pouze bodů rovin, kdežto v mém novém algoritmu je možné odstraňovat body ploch s určenou změnou normál a křivostí.

Vizualizace nalezených bodů je provedena nově upraveným algoritmem prokladu b-spline křivkou. Mnou navržená úprava algoritmu zabezpečuje lepší výsledky prokladu uzavřených b-spline křivek pro potřeby technické praxe.

Výsledné body jsem zobrazil v prostředí PCLVisualizer, který je součástí knihovny PCL.

Navržené metody jsem implementoval v jazyce C++, v prostředí Microsoft Visual Studio Community 2005 (v. 14) za podpory knihovny PCL verze 1.8.1. Provedl jsem analýzu výsledků segmentací, kde byl metodou regresní analýzy vytvořen obecný model pro hledání parametru prahu odchylek normálových vektorů, což je hlavní parametr algoritmu region growing segmentace.

Tato práce je využitelná v oblasti strojírenství, speciálně reverzního inženýrství. V praxi totiž body ostrých hran představují problém a musí se ručně vyhledávat a upravovat. Také skenovací nástroje nedokáží přesně zachytit body přímo na hranách. Vyhledání těchto problematických bodů a jejich proložení křivkou automaticky řeší navržený a implementovaný postup.

V oblasti prokladu b-spline křivkou lze dále pokračovat. Při prokladu b-spline křivkou totiž v některých místech při použití malého počtu řídicích bodů mohou vznikat smyčky či neočekávaná vlnění vlivem vlastností b-spline funkce. Tento jev lze vyřešit vhodným upravením uzlového vektoru.

Další rozšíření práce je možné při propojení nalezené křivky s původními daty pro zpřesnění vytvářeného modelu. Pokud se na proložené křivce vygenerují nové body, pak je lze přidat k původním bodům a zvýšit tak přesnost vytvářeného modelu v těchto místech.

Literatura

Vlastní publikace

[1] Kratochvíl J., Procházková J., Sedlák J., Procházka D. (2017). Automatic sharp feature detection in the point cloud for the reverse engineering applications. *Technical Gazette*. ISSN 1330-3651. IF 0.464 – recenzentní řízení.

[2] Kratochvíl J. (2015). The Overview of Nowadays Approaches on the Automatic Roof Plane Detection and Analysis for Photovoltaic Deployment. *PEFnet 2015, European scientific conference of doctoral students*. Abstracts. Brno, Czech Republic. ISBN 978–80–7509–362–2.

[3] Kratochvíl J. (2015). Umění v matematice, matematika v umění. *Zprávy Vlastivědného muzea v Olomouci* **310**, 24–29.

[4] Procházková J., Kratochvíl J. (2017). Direct point cloud visualization using T–spline with feature detection. *Advances in Intelligent Systems and Computing*. 240–251.

Citovaná literatura

[5] Alshawabkeh Y. (2005). Using Terrestrial Laser Scanning for the 3D Reconstruction of Petra/Jordan. *Photogrammetric Week '05*. 39–47.

[6] Banchoff T., Gaffney T., McCrory C. (1982). Cups of the Gauss Map. *Research Notes in Mathematics*, **55**. Pitman, London.

[7] Bazazian D., Casas J. R., Ruiz-Hidalgo J. (2015). Fast and Robust Edge Extraction in Unorganized Point Clouds. *Digital Image Computing: Techniques and Applications (DICTA)*. 1–8.

[8] Bézier P. E. (1972). *Numerical Control: Matrhematics and Applications*. New York, USA.

[9] Bentey J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*. 509–517.

[10] Blake A., Isard M. (1988). Active Contours. *The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer. New York, USA.

[11] Čermák L. a Hlavička R. (2008). *Numerické metody*. Studijní opora. Brno.

[12] Cox M. G. (1972). The numerical evaluation of B-splines. *Journal of Institute of Mathematics and its Applications*, **10**. 134–149.

[13] De Boor C. (1972). On calculating with B-splines. *Journal of Approximation Theory*, **6**. 50–62.

[14] De Boor C. (1978). A Practical Guide to Splines. *Springer-Verlag*. New York, USA.

[15] Demaris K., Vanderstraeten D., Volodine T., Roose D. (2007). Detection of closed sharp edges in point clouds using normal estimation and graph theory. *Computer Aided Design*, **39**. 267–283.

[16] Documentation – Point Cloud Library (PCL). *The PCD (Point Cloud Data) file format* [online]. Dostupné z: http://pointclouds.org/documentation/tutorials/pcd_file_format.php.

[17] Documentation – Point Cloud Library (PCL). *PCL – Point Cloud Library (PCL)* [online]. Dostupné z: http://pointclouds.org/documentation/.

[18] Enkhbayar P., Damdinsuren S., Osaki M., Matsushima N. (2008). HELFIT: Helix fitting my a total least square method. *Computional Biology and Chemistry*, **32**. 307–310.

[19] Farin G. E. (1983). Algoritms for rational Bézier curves. *Computer Aided Design*, **15** (2). 73–77.

[20] Fischler M. A. a Bolles R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, **24** (6). 381–395.

[21] Forrest A. R. (1990). Interactive interpolation and approximation by Bézier polynomials. *Computer Aided Design*, **22** (9). 527–537.

[22] Frühwirth R, Strandlie A, Waltenberger W. (2002). Helix fitting by an extended Riemann fit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, **490**. 366–378.

[23] Gordon W. J., Reisenfeld R. F. (1974). Bernstein-Bézier methods for the computer--aided design for free-form curves and surfaces. *Journal of the ACM (JACM)*, **21** (2). 293–310. New York.

[24] Gordon W. J., Reisenfeld R. F. (1974). B-spline curves and surfaces. *Computer Aided Geometric Design*. New York. Academic Press. 293–310.

[25] Hildebrandt K., Polthier K., Wardetzky M. (2005). Smooth features lines on surface meshes. *Symposium on geometry processing*. 85–90.

[26] Hoschek J. (1988). Intrinsic parametrization for approximation. *Computer Aided Geometric Design*, **5**. 27–31.

[27] Hoschek J. (1992). Circular splines. *Computer Aided Design*, **24** (11). 611–618.

[28] Hrubý M. *Geografgické Informační Systémy (GIS)*. Studijní opora [online]. Dostupné z: http://perchta.fit.vutbr.cz/vyuka-gis/uploads/a/GIS-final2.pdf.

[29] Chang G., Wu J. (1981). Mathematical foundations of Bézier's technique. *Computer Aided Design*, **13** (3). 133–136.

[30] Chow J. C. K. a Lichti D. D. (2013). Photogrammetric Bundle Adjustment With Self-Calibration of the PrimeSense 3D Camera Technology: Microsoft Kinect. *IEEE Access*, **1**. 465–474.

[31] Christopher J. A., Swanson R., Balwin T. O. (1996). Algorithms for finding the axis of a helix: fast rotational and parametric least–square methods. *Computational Chemistry*, **20**. 339–345.

[32] Kahn P. C. (1989). Defining the axis of a helix. *Computational Chemistry*, **13**. 185–189.

[33] Khoshelham K. a Elberink S. O. (2012). Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications. *Sensors*. 1437–1454.

[34] Kreveld M. v. J. a Overmars M. H. (1991). Divided k–d Trees. *Algorithmica*, **6**. 840–858.

[35] Lancaster P., Salkauskas K. (1986). *Curve and Surface Fitting*. New York: Academic Press.

[36] Lee E. T. Y. (1989). Chosing nodes in parametric curve interpolation. *Computer Aided Design*, **21**. 363–370.

[37] Levoy M. a Whitted T. (2011). The use of points as a display primitive. *Technical Report*. Shanghai, China.

[38] Martišek M. (2002). *Matematické principy grafických systémů*. Littera. ČR.

[39] Maune D. F. (2007). Digital elevation model technologies and applications: the DEM users manual. *American Society for Photogrammetry and Remote Sensing*.

[40] McLachlan A. D. (1979). Gene duplication in the structural evolution of chymotrypsin. *Journal of Molecular Biology*, **128**. 49–79.

[41] Mörwald T., Balzer J., Vincze V. (2016). Modeling connected regions in arbitrary planar point clouds by robust B-spline approximation. *Robotics and Autonomous Systems*, **76**. 141–151.

[42] Nguyen, A., Le B. (2013). 3D point cloud segmentation: A survey. *Robotics, Automation and Mechatronics (RAM)*, 6th IEEE Conference. 225–230.

[43] Miller C. C. (2006). A beast in the field: The Google Maps mashup as GIS/2. *Cartographica: The International Journal for Geographic Information and Geovisualization*, **41** (3). 187–199.

[44] Han X.-F., Jin J. S., Wang W.-J., Jiang W., Xiao L. (2017). A review of algorithms for filtering the 3D point cloud. *Signal Processing: Image Communication*, **57**. 103–112.

[45] Piegl L., Tiller W. (1997). The NURBS Book. 2. vydání. *Springer*. New York.

[46] Object Files (.obj). *Paul Bourke – Personal Pages* [online]. Dostupné z: http://paulbourke.net/dataformats/obj/.

[47] Pauly M., Gross M., Kobbelt L. P. (2002). Efficient simplification of point-sampled surfaces. *Proceedings of the conference on Visualization'02*. IEEE Computer Society. 163–170.

[48] Piegl L. (1991). On NURBS: A Survey. *IEEE Computer Graphics and Applications*, **10** (1). 55–71.

[49] Pomerleau F., Colas F., Siegwart R. (2015). A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics*, **4** (1). 1–104.

[50] PLY – Polygon File Format. *Paul Bourke – Personal Pages* [online]. Dostupné z: http://paulbourke.net/dataformats/ply/.

[51] Quandt R. E. *Some Basic Matrix Theorems*. Princeton University.

[52] Reinsenfeld R. F. (1973). *Applications of B-spline Approximation to Geometric Problems of Computer-Aided Design*. PhD. disertace. Syracuse University.

[53] Rusu R. B. a Cousins S (2011). 3D is here: Point Cloud Library (PCL). *IEEE International Conference on Robotics and Automation (ICRA)*. Computer Science Department, University of North Carolina at Chapel Hill.

[54] Schroeder W. J., Martin K. M., Lorensen W. E. (1996). The design and implementation of an object-oriented toolkit for 3D graphics and visualization. *Proceedings of the 7th conference on Visualization '96*, **93**.

[55] Schwartz B. (2010). LIDAR: Mapping the world in 3D. *Nature Photonics*. London. 429–430.

[56] STL. *Paul Bourke – Personal Pages* [online]. Dostupné z: http://paulbourke.net/dataformats/stl/.

[57] Stoer J., Bulirsch R. (1993). Introduction to Numerical Analysis. *Springer-Verlag*. New York.

[58] Stylianou G. a Farin G. (2003). Crest lines extraction from 3D triangulated meshes. *Hierarchical and geometrical methods in scientific visualization*. 69–81.

[59] Tiller W. (1983). Rational B-splines for curve and surface representation. *IEEE Computer Graphics and Applications*, **3** (6). 61–69.

[60] Unnikrishnan R. (2008). *Statistical approaches to multi-scale point cloud processing*.

[61] Vandergraft J. (1983). *Introduction to Numerical Computations*. New York: Academic Press.

[62] Versprille K. J. (1975). *Computer-Aided Design Applications of the Rational B-spline Approximation Form*. Ph.D. disertace. Syracuse University.

[63] Wavefront .obj file – Wikipedia. [online]. Dostupné z: https://en.wikipedia.org/wiki/Wavefront_.obj_file.

[64] Wang W., Pottmann H., Liu Y. (2006). Fitting B-spline Curves to Point Cloud by Squared Distance Minimization. *HKU CS Tech Report*.

[65] Weber Ch., Hahmann S., Hagen H. (2010). Methods for feature detection in point cloud. *Visualization of Large and Unstructured Data Sets–IRTG Workshop*. 90–99.

[66] Wehra A. a Lohrb U. (1999). Airborne laser scanning–an introduction and overview. *ISPRS Journal of Photogrammetry and Remote Sensing*. 68–82.

[67] Xu, J., Zhou, M., Wu, Z., Shui, W., Ali, S. (2015). Robust surface segmentation and edge feature lines extraction from fractured fragments of relics. *Journal of Computational Design and Engineering*. 79–87.

[68] X3D – Wikipedia. [online]. Dostupné z: https://en.wikipedia.org/wiki/X3D.